

**TUGAS AKHIR - TE 141599**

**PERANCANGAN INSTRUMEN SISTEM NAVIGASI  
DENGAN KALMAN FILTER DAN ALGORITMA  
SMOOTHING SAVITZKY-GOLAY**

Maulana Maliki  
NRP 0711134000018

Dosen Pembimbing  
Ir. Rusdianto Effendi A.K., MT.  
Mochammad Sahal ST., M.Sc.

DEPARTEMEN TEKNIK ELEKTRO  
Fakultas Teknologi Elektro  
Institut Teknologi Sepuluh Nopember  
Surabaya 2018

*[Halaman ini sengaja dikosongkan]*



**TUGAS AKHIR - TE 141599**

**PERANCANGAN INSTRUMEN SISTEM NAVIGASI  
DENGAN KALMAN FILTER DAN ALGORITMA  
*SMOOTHING* SAVITZKY-GOLAY**

Maulana Maliki  
NRP 0711134000018

Dosen Pembimbing  
Ir. Rusdianto Effendi A.K., MT.  
Mochammad Sahal ST., M.Sc.

DEPARTEMEN TEKNIK ELEKTRO  
Fakultas Teknologi Elektro  
Institut Teknologi Sepuluh Nopember  
Surabaya 2018

*[Halaman ini sengaja dikosongkan]*



**FINAL PROJECT - TE 141599**

***DESIGN INSTRUMENT NAVIGATION SYSTEM WITH  
KALMAN FILTER AND SAVITZKY-GOLAY  
SMOOTHING ALGORITHM***

Maulana Maliki  
NRP 0711134000018

*Supervisor*

Ir. Rusdianto Effendi A.K., MT.  
Mochammad Sahal ST., M.Sc.

***DEPARTMENT OF ELECTRICAL ENGINEERING  
Faculty of Electrical Technology  
Institut Teknologi Sepuluh Nopember  
Surabaya 2018***

*[Halaman ini sengaja dikosongkan]*

## PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul

**“Perancangan Instrumen Sistem Navigasi dengan Kalman Filter dan Algoritma *Smoothing* Savitzky-Golay”**

adalah benar benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 3 Juli 2018



Maulana Maliki  
Nrp 07111340000018

*[Halaman ini sengaja dikosongkan]*



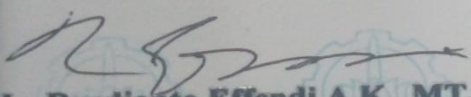
**PERANCANGAN INSTRUMEN SISTEM NAVIGASI  
DENGAN KALMAN FILTER DAN ALGORITMA  
SMOOTHING SAVITZKY-GOLAY**

**TUGAS AKHIR**

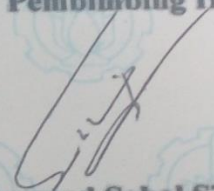
**Diajukan untuk Memenuhi Sebagian persyaratan  
Untuk Memperoleh Gelar Sarjana Teknik  
Pada  
Bidang Studi Teknik Sistem Pengaturan  
Departemen Teknik Elektro  
Institut Teknologi Sepuluh Nopember**

**Menyetujui:**

**Pembimbing I**

  
**Ir. Rusdianto Effendi A K., MT.**  
**NIP 195704241985051001**

**Pembimbing II**

  
**Mochammad Sahal ST., M.Sc.**  
**NIP 197011191998021002**



*[Halaman ini sengaja dikosongkan]*

**PERANCANGAN INSTRUMEN SISTEM NAVIGASI DENGAN  
KALMAN FILTER DAN ALGORITMA *SMOOTHING*  
SAVITZKY-GOLAY**

Maulana Maliki  
07111340000018

Dosen Pembimbing I: Ir. Rusdianto Effendi A K., MT.

Dosen Pembimbing II: Mochammad Sahal ST., M.Sc.

**ABSTRAK**

INS merupakan sistem navigasi berbasis inersia yang paling banyak digunakan, terutama dalam navigasi wahana tak berawak seperti UAV, USV, Rudal, dll. Namun data keluaran yang dibaca oleh Accelerometer dan gyroscope mengandung noise yang berakumulasi seiring waktu dikarenakan proses integral. Untuk mengurangi dan menghilangkan pengaruh noise pada data keluaran sensor navigasi, digunakan filter Kalman sebagai estimator noise. Integrasi menggunakan GPS juga digunakan guna menyediakan sinyal referensi untuk INS, sedangkan algoritma *smoothing* Savitzky-Golay digunakan untuk memperhalus keluaran dari Kalman filter. Hasil dari simulasi menunjukkan bahwa Kalman filter telah berhasil mengestimasi noise dari sinyal INS, meski masih ada mean *error* namun nilainya tergolong kecil, yakni [0.0323, 0.0323, 0.0329] m untuk *error* posisi, [0.0231, 0.0233, 0.0241] untuk *error* kecepatan dan [0.025, 0.0081, 0.0224] untuk *error* orientasi masing-masing pada sumbu [x,y,z].

**Kata Kunci:** INS, Navigasi, GPS, Kalman filter, noise, Savitzky-Golay *Smoothing*

*[Halaman ini sengaja dikosongkan]*

**DESIGN INSTRUMENT NAVIGATION SYSTEM WITH KALMAN  
FILTER AND SAVITZKY-SOLAY SMOOTHING ALGORITHM**

Maulana Maliki  
07111340000018

*Supervisor I:* Ir. Rusdianto Effendi A K., MT.

*Supervisor II:* Mochammad Sahal ST., M.Sc.

**ABSTRACT**

*INS is an inertial-based navigation system which used in so many application such as unmanned vehicle like UAV, USV, missile, etc. But the noise in the output of inertial sensors accelerometer and gyroscope tend to grow without bound, this was caused by integration algorithm in mathematical process in INS. Kalman filter was used as noise estimator to minimize the error. Other navigation system such GPS was also used as aiding to INS for providing a reference signal, so that we can estimate the error model, and lastly the Savitzky-Golay smoothing algorithm was used to smooth and minimize the error even further. From the simulation, the kalman filter succeed to reduce the error caused by noise signal, the error between kalman estimation and signal was [0.0323, 0.0323, 0.0329] m for position, [0.0231, 0.0233, 0.0241] m/s for velocity and [0.025, 0.0081, 0.0224] rad for attitude, all of it in [x,y,z] axis.*

**Key Words:** *INS, Navigation, GPS, Kalman filter, noise, Savitzky-Golay Smoothing*

*[Halaman ini sengaja dikosongkan]*

## Kata Pengantar

Puji syukur penulis panjatkan ke hadirat Allah SWT yang atas berkat rahmat dan karunia-Nya, penulis dapat menyelesaikan laporan Tugas Akhir berjudul **“Perancangan Instrumen Sistem Navigasi dengan Kalman filter dan Algoritma Smoothing Savitzky-Golay”** untuk memenuhi syarat kelulusan pada Bidang Studi Teknik Sistem Pengaturan, Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember, Surabaya.

Buku Laporan tugas akhir ini dapat terselesaikan berkat oleh penulis berkat bimbingan dan bantuan dari Allah SWT. Penulis ingin mengucapkan rasa terima kasih kepada Ibu yang selalu sabar dan mendukung penulis dengan semua bantuan yang dapat diberikan, baik berupa materil maupun non-materil, berkatnya penulis dapat menyelesaikan buku laporan. Terima kasih kepada Bapak Rusdianto Effendi AK MT., dan Bapak Mochammad Sahal ST., M.Sc selaku dosen pembimbing atas masukan, arahan dan ilmu yang disalurkan kepada penulis dalam menyelesaikan buku laporan ini. Dosen-dosen Bidang Studi Teknik Sistem Pengaturan atas ilmu-ilmu yang disalurkan, baik melalui perkuliahan maupun non-perkuliahan. Para senior S2 Teknik Elektro Bidang Studi Teknik Sistem Pengaturan dan keluarga Laboratorium Sistem dan Sibernetik B204. M. Rafif Prasetyo, Febry Angga dan teman-teman yang telah membantu serta berjuang bersama-sama penulis.

Penulis berharap buku laporan ini dapat memberikan manfaat kepada pembaca, baik secara langsung maupun tak langsung. Laporan ini memang masih jauh dari sempurna, oleh karenanya penulis mengharapkan kritik dan saran yang membangun dari pembaca.

Surabaya, 3 Juli 2018

Penulis

*[Halaman ini sengaja dikosongkan]*



## DAFTAR ISI

HALAMAN JUDUL.....	ii
PERNYATAAN KEASLIAN TUGAS AKHIR.....	iii
LEMBAR PENGESAHAN.....	v
ABSTRAK.....	vii
ABSTRACT.....	ix
KATA PENGANTAR.....	xi
DAFTAR ISI.....	xiii
DAFTAR GAMBAR.....	xv
DAFTAR TABEL.....	xix
BAB 1 PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Perumusan Masalah.....	2
1.3 Tujuan.....	2
1.4 Metodologi.....	2
1.5 Sistematika.....	3
1.6 Relevansi.....	3
BAB 2 DASAR TEORI.....	5
2.1 <i>Inertial Navigation System</i> .....	5
2.1.1 <i>Accelerometer</i> .....	7
2.1.2 <i>Gyroscope</i> .....	9
2.2 <i>Global Positioning System</i> .....	10
2.3 Filter Kalman.....	11
2.4 Algoritma <i>Smoothing Savitzky-Golay</i> .....	17
2.5 Model <i>Error</i> .....	22
2.6 Sistem Navigasi Terintegrasi.....	24
2.6.1 Sistem Navigasi Terintegrasi <i>Loosely-Coupled</i> .....	25
2.6.2 Sistem Navigasi Terintegrasi <i>Tightly-Coupled</i> .....	26
2.6.3 Sistem Navigasi Terintegrasi <i>Ultra-Tight</i> .....	27
2.7 Model <i>Error</i> Sistem Terintegrasi.....	28
2.8 Sinkronisasi Data GPS.....	30
BAB 3 PERANCANGAN DAN SIMULASI SISTEM.....	33
3.1 Perancangan Model INS.....	33
3.1.1 Perancangan Model <i>Accelerometer</i> .....	34
3.1.2 Perancangan Model <i>Gyroscope</i> .....	36

3.1.3 Perancangan LPF, HPF dan Model <i>Error</i> .....	41
3.1.4 Perancangan Kalman filter.....	46
3.2 Perancangan GPS.....	49
3.2.1 Perancangan Ekstrapolasi.....	51
3.3 Perancangan Sistem Terintegrasi.....	53
BAB 4 HASIL SIMULASI DAN ANALISA.....	57
4.1 Hasil Simulasi INS dengan Filter Kalman.....	57
4.2 Hasil Simulasi Ekstrapolasi Data GPS.....	65
4.3 Hasil Simulasi Integrasi INS-GPS dengan Filter Kalman.....	67
4.4 Hasil Simulasi Integrasi INS-GPS dengan Filter Kalman dan Algoritma <i>Smoothing</i> Savitzky-Golay.....	75
BAB 5 KESIMPULA DAN SARAN.....	81
5.1 Kesimpulan.....	81
5.2 Saran.....	81
DAFTAR PUSTAKA.....	83
LAMPIRAN.....	85
Lampiran 1.....	85
Lampiran 2.....	86
Lampiran 3.....	88
Lampiran 4.....	89

## DAFTAR GAMBAR

Gambar 2.1 Susunan komponen pada INS tipe <i>gimbal</i> (a) dan <i>strapdown</i> (b).....	5
Gambar 2.2 <i>Inertial Measurement Unit</i> .....	6
Gambar 2.3 <i>Accelerometer</i> pada titik <i>equilibrium</i> .....	7
Gambar 2.4 Pembacaan percepatan pada <i>Accelerometer</i> .....	7
Gambar 2.5 Pembacaan percepatan gravitasi pada <i>Accelerometer</i> .....	8
Gambar 2.6 Koordinat <i>body frame</i> .....	9
Gambar 2.7 Ilustrasi <i>Roll, Pitch, Yaw</i> .....	10
Gambar 2.8 Konsep dasar GPS .....	11
Gambar 2.9 <i>Smoothing</i> data dengan polinomial orde $d = 0,1,2$ .....	17
Gambar 2.10 <i>Gain</i> Butterworth untuk orde 1 sampai 5.....	24
Gambar 2.11 Sistem Navigasi Terintegrasi <i>Loosely-Coupled</i> .....	26
Gambar 2.12 Sistem Navigasi Terintegrasi <i>Tightly-Coupled</i> .....	27
Gambar 2.13 Sistem Navigasi Terintegrasi <i>Ultra-Tight</i> .....	28
Gambar 2.14 Diagram blok proses ekstrapolasi .....	30
Gambar 3.1 Perancangan <i>Accelerometer</i> (a) dan <i>Gyroscope</i> (b) INS....	33
Gambar 3.2 Diagram Simulink Perancangan Simulasi <i>Accelerometer</i> ....	34
Gambar 3.3 Blok Subsistem <i>Accelerometer</i> pada Simulasi INS.....	35
Gambar 3.4 Blok Parameter dari <i>Band-Limited White Noise</i> pada <i>Accelerometer</i> .....	35
Gambar 3.5 Blok Simulink Perancangan Simulasi <i>Gyroscope</i> .....	36
Gambar 3.6 Blok Subsistem <i>Gyroscope</i> pada Simulasi INS.....	36
Gambar 3.7 Blok Parameter dari <i>Band-Limited White Noise</i> pada <i>Gyroscope</i> .....	37
Gambar 3.8 Blok Parameter untuk data <i>roll</i> ( $\Phi, \varphi$ ).....	38
Gambar 3.9 Blok Parameter untuk data <i>pitch</i> ( $\theta, \theta$ ).....	39
Gambar 3.10 Blok Parameter untuk data <i>yaw</i> ( $\Psi, \psi$ ).....	40
Gambar 3.11 Diagram Blok perancangan Model <i>Error</i> .....	41
Gambar 3.12 Diagram Simulink untuk LPF.....	42
Gambar 3.13 Diagram Simulink untuk HPF.....	42
Gambar 3.14 Bagian dalam blok subsistem LPF.....	42
Gambar 3.15 Bagian dalam blok sistem HPF.....	43
Gambar 3.16 Nilai frekuensi <i>cut-off</i> dari LPF dan HPF.....	44

Gambar 3.17 Diagram Simulink Model <i>Error</i> .....	45
Gambar 3.18 Diagram blok proses estimasi filter Kalman.....	46
Gambar 3.19 Blok Simulink Kalman filter.....	47
Gambar 3.20 Diagram simulink Model INS dengan Kalman filter.....	48
Gambar 3.21 Diagram blok perancangan simulasi GPS.....	50
Gambar 3.22 Blok Simulink perancangan GPS.....	50
Gambar 3.23 Bagian dalam blok subsistem Model GPS.....	51
Gambar 3.24 Blok Simulink <i>Zero-Order Hold</i> .....	52
Gambar 3.25 Diagram Simulink Ekstrapolasi.....	52
Gambar 3.26 Diagram Simulink Model GPS dan Ekstrapolasi.....	53
Gambar 3.27 Diagram blok sistem terintegrasi INS-GPS <i>loosely-coupled</i> .....	53
Gambar 3.28 Diagram blok simulink sistem terintegrasi INS-GPS.....	55
Gambar 4.1 Data percepatan tanpa <i>noise</i> .....	57
Gambar 4.2 Data orientasi tanpa <i>noise</i> .....	58
Gambar 4.3 Data kecepatan sudut tanpa <i>noise</i> .....	58
Gambar 4.4 Data Percepatan + <i>noise</i> .....	59
Gambar 4.5 Data kecepatan sudut + <i>noise</i> .....	59
Gambar 4.6 Perbandingan antara estimasi posisi dan posisi asli.....	60
Gambar 4.7 Perbandingan antara estimasi <i>velocity</i> dengan <i>velocity</i> asli.....	60
Gambar 4.8 Perbandingan Sudut <i>Roll</i> .....	61
Gambar 4.8 Perbandingan Sudut <i>Pitch</i> .....	61
Gambar 4.10 Perbandingan Sudut <i>Yaw</i> .....	62
Gambar 4.11 <i>Error</i> Posisi.....	63
Gambar 4.12 <i>Error Velocity</i> .....	63
Gambar 4.13 <i>Error</i> Orientasi.....	64
Gambar 4.14 Plot hasil ekstrapolasi data posisi GPS.....	66
Gambar 4.15 Plot hasil ekstrapolasi data <i>velocity</i> GPS.....	66
Gambar 4.16 Data percepatan tanpa <i>noise</i> .....	67
Gambar 4.17 Data orientasi tanpa <i>noise</i> .....	68
Gambar 4.18 Data kecepatan sudut tanpa <i>noise</i> .....	68
Gambar 4.19 Data percepatan + <i>noise</i> .....	69
Gambar 4.20 Data kecepatan sudut + <i>noise</i> .....	69
Gambar 4.21 Data posisi GPS.....	70
Gambar 4.22 Data <i>velocity</i> GPS.....	70

Gambar 4.23 Perbandingan posisi.....	71
Gambar 4.24 Perbandingan kecepatan ( <i>velocity</i> ).....	71
Gambar 4.25 Perbandingan orientasi.....	72
Gambar 4.26 <i>Error</i> posisi.....	73
Gambar 4.27 <i>Error</i> kecepatan.....	73
Gambar 4.28 <i>Error</i> orientasi.....	74
Gambar 4.29 Perbandingan posisi antara est. kalman dengan <i>smoothing</i> S-G untuk percobaan 1.....	76
Gambar 4.30 Perbandingan velocity antara est. kalman dengan <i>smoothing</i> S-G untuk percobaan 1.....	76
Gambar 4.31 Perbandingan orientasi antara est. kalman dengan <i>smoothing</i> S-G untuk percobaan 1.....	77
Gambar 4.32 Perbandingan posisi antara est. kalman dengan <i>smoothing</i> S-G untuk percobaan 2.....	77
Gambar 4.33 Perbandingan velocity antara est. kalman dengan <i>smoothing</i> S-G untuk percobaan 2.....	78
Gambar 4.34 Perbandingan orientasi antara est. kalman dengan <i>smoothing</i> S-G untuk percobaan 2.....	78

*[Halaman ini sengaja dikosongkan]*

## DAFTAR TABEL

Tabel 2.1 Perbandingan INS tipe <i>gimbal</i> dan tipe <i>strapdown</i> .....	6
Tabel 2.2 Tabel Normalisasi Polinomial Butterworth.....	24
Tabel 2.3 Komparasi nilai bias INS.....	29
Tabel 3.1 Nilai frekuensi <i>cut-off</i> dari LPF dan HPF.....	43
Tabel 4.2 Hasil Statistik <i>error</i> .....	64
Tabel 4.5 Statistik <i>error</i> untuk hasil simulasi sistem INS-GPS terintegrasi.....	74
Tabel 4.6 Perbandingan <i>error</i> statistik antara hasil Kalman dengan <i>smoothing</i> Savitzky-Golay.....	79

*[Halaman ini sengaja dikosongkan]*



## BAB 1 PENDAHULUAN

### 1.1 Latar Belakang

Sistem Navigasi Inersia (*Inertial Navigation System*) merupakan sistem navigasi yang mengubah data hasil pembacaan sensor navigasi berupa percepatan linier (didapat dari *accelerometer*) dan kecepatan sudut (didapat dari *gyroscope*) untuk mendapatkan data berupa posisi, kecepatan linier, dan orientasi. Hasil pengukuran yang didapat dari kedua sensor tersebut disebut dengan IMU atau *Inertial Measurement Unit* (Unit Pengukuran Inersia). Sistem navigasi inersia digunakan dalam berbagai aplikasi navigasi, termasuk pada kapal tanpa awak, rudal, pesawat ruang angkasa dan sebagainya.

Permasalahan yang muncul pada sistem navigasi inersia adalah *error* pada data IMU karena berbagai hal, seperti *noise* dikarenakan getaran pada *body* kendaraan, efek gravitasi, serta offset pada sensor. *Error* ini akan bertambah besar seiring berjalannya waktu akibat proses integral yang ada pada INS. Untuk itu, diperlukan sebuah metode yang mampu mengurangi/menghilangkan *noise* yang menjadi penyebab *error*. Filter Kalman merupakan salah satu metode yang umum digunakan dalam mengestimasi nilai *error* yang dihasilkan oleh data IMU, sehingga nantinya nilai estimasi *error* tersebut dapat menghasilkan solusi navigasi yang lebih akurat.

Salah satu cara yang sering digunakan adalah dengan merancang sebuah sistem terintegrasi antara INS dan GPS (*Global Positioning System*). GPS merupakan sistem navigasi berbasis satelit yang dikembangkan oleh Departemen Pertahanan Amerika Serikat di bawah program satelit NAVSTAR. GPS mampu menghasilkan data posisi serta kecepatan linier dengan *error* yang kecil, sehingga menjadikan sistem navigasi berbasis satelit ini sebagai penyedia sinyal referensi bagi INS. Ada tiga macam sistem integrasi antara INS dan GPS, yaitu *loosely coupled*, *tightly coupled*, dan *ultra-tight coupled*. Pada buku ini, penulis memilih untuk menggunakan integrasi *loosely coupled*.

Kemudian, untuk lebih meningkatkan hasil dari sistem terintegrasi antara INS dan GPS, dilakukan proses *smoothing*. Metode *smoothing* yang dipakai pada tugas akhir ini adalah metode *smoothing* Savitzky-Golay, yaitu metode *smoothing* yang bertujuan untuk meningkatkan rasio *signal-to-noise* (SNR) tanpa terlalu merusak/mendistorsi sinyal. Hal ini

dapat dicapai melalui proses konvolusi, yaitu dengan melakukan *fitting* pada data poin dengan polinomial orde rendah.

## 1.2 Perumusan Masalah

Sistem navigasi inersia menggunakan data yang didapat dari IMU berupa percepatan dan kecepatan sudut untuk mendapatkan posisi, kecepatan linier, dan orientasi dari benda bergerak. Namun, karena adanya gangguan berupa *noise* karena getaran, pengaruh gravitasi, dan *offset* menyebabkan *error* yang terus bertambah seiring waktu.

## 1.3 Tujuan

Merancang sistem navigasi inersia terintegrasi dengan GPS ditambah dengan filter Kalman serta algoritma *smoothing* Savitzky-Golay untuk menghilangkan *noise* dan memperkecil *error* dari data navigasi IMU.

## 1.4 Metodologi

Metodologi yang digunakan pada pengerjaan tugas akhir ini diawali dengan studi literatur terkait topik yang dipilih. Sumber referensi diambil dari buku ilmiah, makalah ilmiah, serta artikel ilmiah dari sumber yang terpercaya dan telah melalui *peer review* dari kalangan ilmiah. Studi literatur ini menjadi pondasi untuk melakukan desain / perancangan sistem navigasi inersia, model GPS, filter Kalman dan algoritma *smoothing*.

Setelah didapat dinamika INS dan GPS dari studi literatur, dilakukan perancangan model INS dan GPS, serta model *noise* yang nantinya akan digunakan pada INS. Kemudian, dilakukan perancangan model integrasi antara INS dan GPS. Selisih posisi dan kecepatan dari kedua sistem tersebut beserta *error* orientasi diturunkan menjadi model *error*, yang nantinya akan diestimasi menggunakan filter Kalman untuk mendapatkan estimasi *error*, barulah setelah itu dilakukan *smoothing* menggunakan algoritma *smoothing* Savitzky-Golay.

Setelah simulasi dari model yang dibuat telah selesai dilakukan, dilakukan penyusunan buku laporan tugas akhir yang berisi tentang seluruh kegiatan yang telah dilakukan.

## **1.5 Sistematika**

Penulisan laporan tugas akhir ini mengikuti sistematika sebagai berikut.

### **BAB 1 Pendahuluan**

Bab ini menguraikan tentang latar belakang dari pemilihan judul tugas akhir. Permasalahan yang ada, tujuan dari pelaksanaan tugas akhir, metodologi penelitian, sistematika penulisan, dan relevansi dari tugas akhir ini.

### **BAB 2 Dasar Teori**

Pada bab ini dijabarkan teori-teori penunjang dan teori-teori dasar yang didapatkan dari tinjauan pustaka yang menjadi acuan dalam melaksanakan tugas akhir ini.

### **BAB 3 Perancangan Simulasi Sistem**

Dalam bab ini, dilakukan perancangan model simulasi yang digunakan dalam tugas akhir, yakni model INS dan GPS, model integrasi sistem, model *error*, filter Kalman dan algoritma *smoothing*.

### **BAB 4 Hasil Simulasi dan Analisa**

Pada bab ini akan ditampilkan hasil simulasi dari pengujian model sistem yang telah dibuat. Hasil dari simulasi kemudian akan di analisa.

### **BAB 5 Kesimpulan dan Saran**

Bab ini berisikan kesimpulan yang didapat dari analisa hasil simulasi, serta pemberian saran untuk penelitian berikutnya mengenai topik terkait.

## **1.6 Relevansi**

Hasil dari tugas akhir ini diharapkan mampu memberi manfaat bagi khalayak umum, salah satunya adalah dengan menjadi referensi mengenai perancangan sistem navigasi inersia.

*[Halaman ini sengaja dikosongkan]*

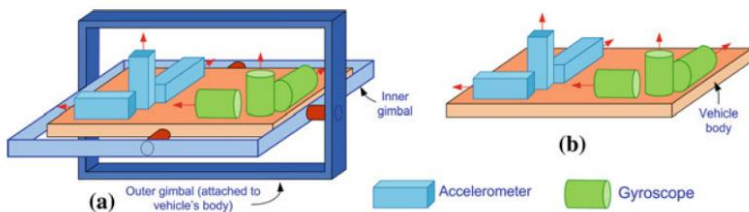
## BAB 2

### DASAR TEORI

#### 2.1 *Inertial Navigation System*

Sistem navigasi inersia (*Inertial Navigation System* / INS) adalah sistem navigasi otonom yang menyediakan informasi berupa posisi, kecepatan dan orientasi dari benda bergerak berdasarkan hasil pengukuran sensor inersia berupa *accelerometer* dan *gyroscope* dan penerapan prinsip *dead reckoning* (DR) [1]. *Dead reckoning* adalah penentuan dari posisi kendaraan saat ini berdasarkan pengetahuan berupa informasi tentang posisi kendaraan sebelumnya, percepatan yang terukur oleh sensor serta kecepatan sudut. Integral pertama dari hasil pembacaan sensor *accelerometer* akan menghasilkan data berupa kecepatan/*velocity* benda dan integral kedua akan menghasilkan data berupa posisi benda. Sementara data kecepatan angular akan memberikan informasi berupa orientasi dari benda / kendaraan tersebut dalam bentuk *roll*, *pitch* dan *yaw*.

Terdapat dua jenis implementasi INS, yakni *platform* stabil yang dikenal dengan sistem gimbal (gambar 2.1 (a)), dan sistem *strapdown* (gambar 2.1 (b)). Pada sistem *platform* stabil, sensor-sensor inersia diletakkan pada sebuah gimbal yang selalu dilakukan *alignment* sedemikian rupa sehingga tiga dari masing-masing sensor tersebut membaca data untuk setiap sumbu kartesian 3 dimensi. Karena sistem gimbal merupakan sistem yang kompleks secara mekanik dan mahal, aplikasinya pada dunia navigasi terbatas.



**Gambar 2.1** Susunan komponen pada INS tipe *gimbal* (a) dan *strapdown* (b)

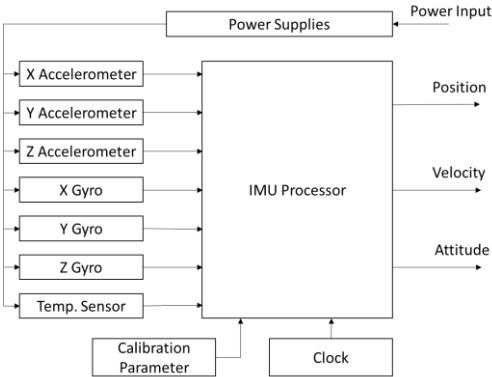
Perkembangan teknologi elektronika melahirkan sistem *strapdown*. Pada sistem ini, sensor-sensor inersia tertanam pada *body* kendaraan dan *gimbal* digantikan oleh komputer yang memperhitungkan serta

mensimulasikan rotasi dan pergerakan yang terjadi pada kendaraan. Sistem *strapdown* lebih diminati karena keandalan, fleksibilitas, penggunaan daya yang kecil, bobotnya yang ringan serta harganya yang lebih murah dari sistem gimbal. Tabel 2.1 merupakan tabel perbandingan INS teknologi gimbal dan *strapdown*.

**Tabel 2.1** Perbandingan INS tipe *gimbal* dan *strapdown*

Karakteristik	INS tipe <i>Gimbal</i>	INS tipe <i>Strapdown</i>
Ukuran	Besar	relative kecil
Berat	Berat	relative lebih ringan
Performa	Performansi yang lebih baik	Akurasi tinggi
<i>Robust</i>	Keandalan tinggi namun kurang memiliki ketahanan terhadap guncangan dan getaran	Keandalan tinggi, tahan terhadap guncangan dan getaran

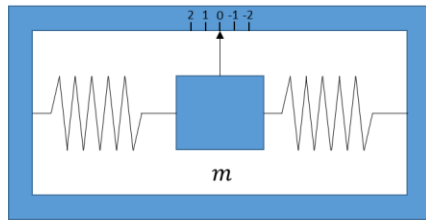
Dalam INS terdapat unit pengukuran inersia (IMU) yang melakukan pengolahan awal data-data yang didapat dari sensor inersia (*accelerometer* dan *gyroscope*). Data yang didapatkan *accelerometer* berupa percepatan sedangkan data yang dibaca oleh *gyroscope* berupa kecepatan sudut. Keduanya diperlukan dalam menentukan posisi suatu benda bergerak. Untuk setiap sumbu, terpasang sensor *accelerometer* dan *gyroscope*, yang berarti akan ada 3 sensor *accelerometer* dan 3 sensor *gyroscope* mengukur data untuk setiap sumbu kartesian.



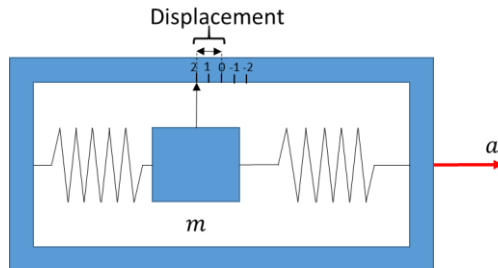
**Gambar 2.2** *Inertial Measurement Unit (IMU)*

### 2.1.1 Accelerometer

*Accelerometer* merupakan sensor inersia yang membaca data berupa percepatan yang dialami kendaraan atau benda bergerak yang diukur. Seperti yang terlihat pada gambar 2.3 , *accelerometer* terdiri atas *proof mass*,  $m$  , yang terhubung dengan sepasang pegas. Apabila terjadi percepatan pada benda dalam satu sumbu, maka *proof mass* akan bergerak dari titik *equilibrium*, pergerakan/pergeseran *proof mass* dari titik keseimbangan akan terbaca oleh *pick off* dan terukur pada skala yang telah ditentukan. Gambar 2.4 menunjukkan pergerakan/pergeseran *proof mass* dari titik keseimbangan dikarenakan adanya percepatan yang terjadi.



**Gambar 2.3** Accelerometer pada titik *equilibrium*



**Gambar 2.4** Pembacaan percepatan pada Accelerometer

Sementara, ketika sensor *accelerometer* dipasang pada keadaan diam pada sumbu-z atau sumbu vertikal, pengaruh percepatan gravitasi akan menyebabkan pergeseran *proof mass* ke arah bawah yang nilainya setara dengan percepatan gravitasi yang dialami sensor. Pada sumbu-z, hasil keluaran dari *accelerometer* adalah sebagai berikut:

$$f = a - g \quad (2.1)$$

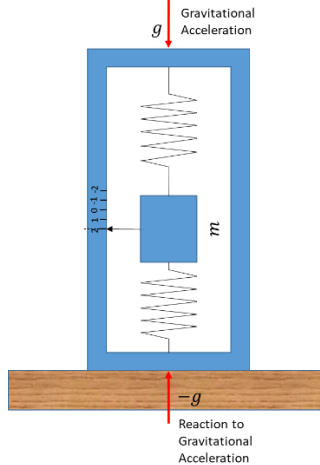
Dimana

$f$  adalah gaya spesifik (*specific force*) yang terbaca oleh *accelerometer*

$a$  adalah percepatan yang sebenarnya, sedangkan

$g$  adalah percepatan gravitasi yang nilainya  $\pm 9.8 \text{ m/s}^2$ .

Dari persamaan 2.1, jika *accelerometer* diletakkan sejajar dengan sumbu-z dalam keadaan diam, maka nilai dari  $a$  adalah sama dengan nol dan menyebabkan keluaran dari *accelerometer* adalah  $f = -g$ . Namun, pada praktiknya, nilai keluaran dari *accelerometer* pada sumbu-z dalam keadaan diam adalah sama dengan  $g$ . Untuk menerangkan hal ini, gaya spesifik yang terbaca oleh *accelerometer* sebenarnya adalah gaya reaksi yang terjadi akibat percepatan gravitasi yang arahnya adalah berlawanan dengan percepatan gravitasi itu sendiri (menuju ke arah atas), yaitu negatif dari  $g$ . Hal ini ditunjukkan pada gambar 2.5.



**Gambar 2.5** Pembacaan percepatan gravitasi pada *Accelerometer*

Sementara, percepatan yang terjadi pada bidang horizontal (sumbu-x dan sumbu-y) akan dipengaruhi oleh gaya sentripetal. Gaya sentripetal ini diakibatkan oleh bumi yang terus berotasi, atau disebut dengan efek koriolis. Sehingga, secara keseluruhan, output dari *accelerometer* untuk ketiga sumbu (x, y, dan z) adalah sebagai berikut:

$$\mathbf{f} = \mathbf{a} - \mathbf{\bar{g}} \quad (2.2)$$

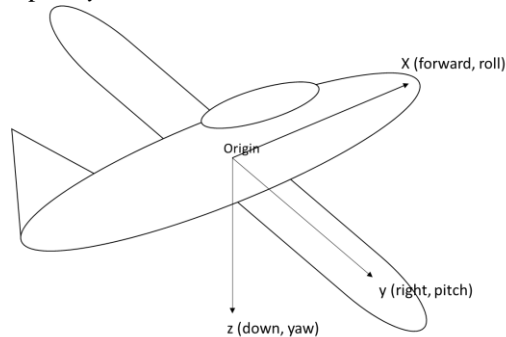
Dimana  $\mathbf{f}$  adalah vektor gaya spesifik, yang mana elemennya merupakan gaya spesifik yang terjadi untuk setiap sumbu x, y, dan z.  $\mathbf{a}$  adalah percepatan yang terjadi pada benda untuk setiap sumbu x, y, dan z, serta  $\mathbf{\bar{g}}$  adalah vektor medan gravitasi yang terjadi karena efek percepatan



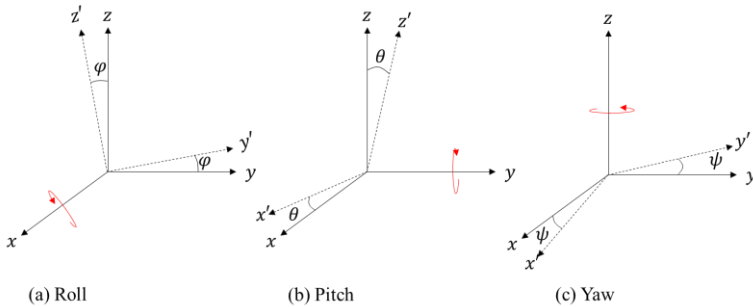
gravitasi dan efek koriolis. Terdapat dua jenis sensor *accelerometer*, yakni *accelerometer* tipe pendulum dan *accelerometer* tipe *vibrating-beam*.

### 2.1.2 Gyroscope

*Gyroscope* merupakan sensor yang membaca perubahan sudut pada benda/kendaraan terhadap sumbu kartesian 3 dimensi. Apabila dibentuk sebuah sumbu kartesian tiga dimensi pada benda/kendaraan dengan titik pusat berada pada pusat massa kendaraan, dengan sumbu-x adalah arah depan (*forward*), sumbu-y adalah arah kanan kendaraan (*right*) dan sumbu-z adalah arah bawah (*down*) menciptakan sistem kartesian dengan kaedah tangan kanan seperti yang terlihat pada gambar 2.6. Adanya pergerakan *angular* terhadap sumbu-x menyebabkan perubahan sudut yang disebut dengan sudut *roll*, sementara perubahan sudut yang diakibatkan karena pergerakan *angular* pada sumbu-y disebut dengan sudut *pitch* dan pergerakan *angular* pada sumbu-z menyebabkan terjadinya perubahan sudut yang disebut dengan sudut *yaw*. Gambar 2.7 mengilustrasikan tentang pergerakan angular dan sudut *roll*, *pitch*, serta *yaw* yang terjadi pada kendaraan. *Gyroscope* membaca perubahan sudut-sudut yang terjadi seraya kendaraan bergerak, perubahan sudut tersebut terjadi terhadap *body* kendaraan.



**Gambar 2.6** Koordinat *body frame*



**Gambar 2.7** Ilustrasi *Roll, Pitch, Yaw*

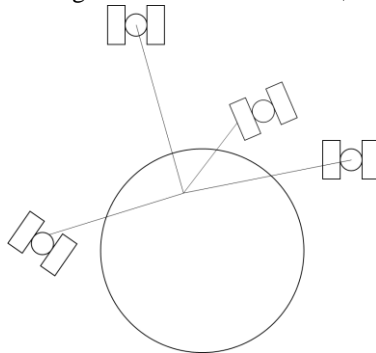
Ada tiga jenis *gyroscope*, yakni *gyroscope* tipe *spinning-mass*, *gyroscope* tipe optik dan *gyroscope* tipe vibrasi.

## 2.2 Global Positioning System

*Global Positioning System*, atau yang biasa disebut GPS merupakan sistem navigasi berbasis satelit yang dikembangkan oleh Departemen Pertahanan Amerika Serikat di bawah program NAVSTAR (*Navigation by Satellite Ranging and Timing*). GPS merupakan sistem navigasi berbasis satelit yang menghasilkan solusi navigasi berupa posisi dimensi tiga ( $x, y, z$ ) menggunakan sinyal radio yang dipancarkan melalui satelit yang mengorbit [3].

GPS merupakan salah satu bagian dari sistem navigasi berbasis satelit. Sistem navigasi berbasis satelit yang pertama kali dikembangkan adalah Sistem transit milik Angkatan Laut Amerika Serikat (*U.S. Navy's Transit System*) [2]. Pengembangan dilakukan pada tahun 1958 dengan satelit percobaan pertama diluncurkan pada tahun 1961 dan sistem beroperasi secara optimal pada tahun 1964. Sistem navigasi ini mulai terbuka untuk umum pada tahun 1967. Sedangkan pengembangan dari GPS dimulai pada tahun 1973. Satelit prototip pertama yang beroperasi diluncurkan pada tahun 1978 dan kemampuan operasional awal (*Initial Operational Capability/IOC*) dari sistem GPS penuh diumumkan pada tahun 1993. Meskipun dikembangkan untuk keperluan militer, kini GPS dapat digunakan untuk kepentingan umum. Sistem Navigasi Satelit Global (*Global Navigation Satellite System* atau disingkat menjadi GLONASS) adalah sistem militer yang dioperasikan oleh Rusia dan

dikembangkan bersamaan dengan GPS, satelit pertamanya diluncurkan pada tahun 1982. Sistem navigasi berbasis satelit yang ketiga, Galileo, dikembangkan oleh Uni Eropa dan negara-negara yang menjadi bagian dari proyek tersebut. satelit pertama diluncurkan pada tahun 2005 dan IOC sudah diumumkan pada 21 Oktober 2011 lalu, sementara kemampuan operasional penuh (*Full Operational Capability/FOC*) diharapkan dapat terpenuhi pada tahun 2019 [6]. Proyek ini dinamai berdasarkan nama seorang ilmuwan astronomi Italia, Galelio Galilei.



**Gambar 2.8** Konsep dasar GPS

Gambar 2.8 menunjukkan konsep dasar dari cara kerja satelit. GPS, GLONASS dan Galileo masing-masing didesain untuk menyusun sebuah konstelasi yang terdiri atas 24 atau lebih satelit yang mengorbit bumi pada radius antara 25,000 hingga 30,000 km dari pusat bumi, untuk memastikan bahwa sinyal dari sedikitnya empat satelit dapat diterima receiver di permukaan bumi manapun. Pada prakteknya, jumlah satelit yang terpantau oleh *receiver* bisa lebih dari empat, hal ini membuat akurasi dari estimasi posisi dapat lebih baik dan konsisten saat sistem navigasi digunakan. Namun, jumlah satelit yang terlihat oleh *receiver* bisa juga kurang dari 4, hal ini dikarenakan beberapa hal, seperti gedung-gedung atau pepohonan tinggi yang menghalangi sinyal transmisi sampai pada *receiver*. Kondisi ini merupakan salah satu gangguan yang terjadi pada GPS dan dapat menurunkan akurasi GPS dalam mengestimasi jarak dan posisi *receiver* di permukaan bumi.

### 2.3 Filter Kalman

Pada tahun 1940, N. Wiener menggagas sebuah teori filtrasi modern yang didasarkan pada cara meminimisasi *mean-square error*, sehingga cabang dari teori filtrasi ini kadang disebut sebagai “*linear time-domain minimum mean-square error filtering*”. Pokok masalah yang dibahas dalam penelitian yang dikerjakan oleh Wiener adalah bagaimana memisahkan sinyal asli dari sinyal yang merupakan kombinasi aditif antara sinyal asli dan *noise* [5].

Sedangkan tahun 1960, Rudolph E. Kalman mencoba mengatasi masalah yang dihadapi oleh Wiener, namun dalam papernya ia menganggap *noise* dalam pengukuran sebagai sebuah sekuen diskrit dibandingkan dengan sinyal waktu kontinyu. Ia juga menunjukkan penyelesaian dalam *state-space*. Para insinyur, terutama dalam bidang navigasi, menemukan bahwa Kalman filter sebagai solusi praktik untuk digunakan dalam masalah filtrasi yang tidak bisa diselesaikan oleh Wiener filter. Percepatan perkembangan komputer pada tahun 1960 juga menambah popularitas Kalman filter sebagai metode untuk memisahkan sinyal dengan *noise*. Bahkan hingga saat ini, filter Kalman masih digunakan dalam penyelesaian permasalahan filtrasi.

Untuk melakukan estimasi, filter Kalman memerlukan dua buah persamaan matematika, yaitu model sistem dan model pengukuran.

$$\mathbf{x}_{k+1} = \boldsymbol{\phi}_k \mathbf{x}_k + \mathbf{w}_k \quad (2.3)$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \quad (2.4)$$

Dimana (2.3) merupakan persamaan model sistem waktu diskrit dan (2.4) merupakan persamaan model pengukuran dengan:

$\mathbf{x}_k = (n \times 1)$  vektor state proses saat waktu  $t_k$ .

$\boldsymbol{\phi}_k = (n \times n)$  matriks transisi state antara  $\mathbf{x}_k$  dengan  $\mathbf{x}_{k+1}$ .

$\mathbf{w}_k = (n \times 1)$  vektor kovarian dari *noise* sistem (asumsi *white noise*).

$\mathbf{z}_k = (m \times 1)$  vektor pengukuran pada waktu  $t_k$ .

$\mathbf{H}_k = (m \times n)$  Matriks transisi state antara pengukuran  $\mathbf{z}_k$  dengan vektor state  $\mathbf{x}_k$ .

$\mathbf{v}_k = (m \times 1)$  vektor kovarian *error* pengukuran (asumsi *white noise*) dan tidak berkorelasi dengan  $\mathbf{w}_k$ .

Matriks kovarian untuk  $\mathbf{w}_k$  dan  $\mathbf{v}_k$  adalah sebagai berikut:

$$E[\mathbf{w}_k \mathbf{w}_i^T] = \begin{cases} \mathbf{Q}_k, & i = k \\ 0, & i \neq k \end{cases} \quad (2.5)$$

$$E[\mathbf{v}_k \mathbf{v}_i^T] = \begin{cases} \mathbf{R}_k, & i = k \\ 0, & i \neq k \end{cases} \quad (2.6)$$

$$E[\mathbf{w}_k \mathbf{v}_i^T] = 0, \text{ untuk semua } k \text{ dan } i. \quad (2.7)$$

Pada poin ini diasumsikan bahwa kita memiliki estimasi awal dari proses pada waktu ke  $t_k$ , dan bahwa estimasi ini didasarkan pada proses yang terjadi sebelum (*prior to*) proses waktu  $t_k$ . Estimasi sementara ini dinotasikan dengan  $\hat{\mathbf{x}}_k^-$  dengan berdasar pada aturan bahwa estimasi sementara ini merupakan estimasi terbaik sebelum diasimilasikan dengan data pengukuran pada waktu  $t_k$ . Diasumsikan bahwa matriks kovarian *error* juga berhubungan dengan  $\hat{\mathbf{x}}_k^-$ . Dengan demikian, estimasi *error* dapat didefinisikan sebagai berikut:

$$\mathbf{e}_k^- = \mathbf{x}_k - \hat{\mathbf{x}}_k^- \quad (2.8)$$

Dan matriks kovarian *error* adalah

$$\mathbf{P}_k^- = E[\mathbf{e}_k^- \mathbf{e}_k^{-T}] = E[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^T] \quad (2.9)$$

Pada berbagai kasus, proses estimasi dimulai tanpa data pengukuran sebelumnya, yang berarti mean dari proses adalah nol dan estimasi awal adalah nol, serta matriks kovarian *error* adalah matriks kovarian dari  $\mathbf{x}$  itu sendiri.

Setelah selesai mengasumsikan estimasi sementara  $\hat{\mathbf{x}}_k^-$ , digunakan data pengukuran  $\mathbf{z}_k$  untuk meningkatkan estimasi sementara. Dipilih sebuah blending linier dari pengukuran *bernoise* dan estimasi sementara berdasarkan pada persamaan

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-) \quad (2.10)$$

Dimana

$\mathbf{x}_k$  = Estimasi terbaru

$\mathbf{K}_k$  = Blending factor (belum ditentukan)

Pada poin ini, masalah yang harus diselesaikan adalah mencari nilai  $\mathbf{K}_k$  yang menghasilkan estimasi terbaru yang optimal. Seperti pada Wiener filter, digunakan minimum mean-square *error* sebagai indeks performansi. Pertama-tama, dibentuk sebuah persamaan yang mengekspresikan matriks kovarian *error* yang berasosiasi dengan estimasi terbaru sebagai berikut

$$\mathbf{P}_k = E[\mathbf{e}_k \mathbf{e}_k^T] = E[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T] \quad (2.11)$$

Berikutnya, substitusi persamaan 2.4 ke persamaan 2.10 dan kemudian substitusi hasil tersebut sebagai ekspresi dari  $\hat{\mathbf{x}}_k$  ke dalam persamaan 2.11 menghasilkan

$$\mathbf{P}_k = E\{[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-) - \mathbf{K}_k(\mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-)][(\mathbf{x}_k - \hat{\mathbf{x}}_k^-) - \mathbf{K}_k(\mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-)]^T\} \quad (2.12)$$

$(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)$  merupakan *error* estimasi sebelumnya yang sama sekali tak berkorelasi dengan *error* pengukuran saat ini,  $\mathbf{v}_k$ , dapat dituliskan bahwa  $\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T$  (2.13) Perlu diketahui bahwa persamaan 2.13 adalah persamaan umum yang mengekspresikan matriks *error* kovarian terbaru, dan berlaku untuk nilai  $\mathbf{K}_k$  manapun, suboptimal atau tidak.

Kembali ke persoalan optimasi, kita ingin menemukan nilai  $\mathbf{K}_k$  yang meminimisasi setiap elemen diagonal dari  $\mathbf{P}_k$ , karena elemen-elemen tersebut merepresentasikan varian estimasi *error* untuk setiap elemen-elemen dari vektor state yang diestimasi. Optimasi menggunakan pendekatan differensial, untuk melakukannya diperlukan dua formula penurunan matriks, yaitu

$$\frac{d[\text{trace}(\mathbf{AB})]}{d\mathbf{A}} = \mathbf{B}^T \quad (\mathbf{AB} \text{ harus matriks persegi}) \quad (2.14)$$

$$\frac{d[\text{trace}(\mathbf{ACA}^T)]}{d\mathbf{A}} = 2\mathbf{AC} \quad (\mathbf{C} \text{ harus simetris}) \quad (2.15)$$

Dimana turunan dari skalar terhadap matriks didefinisikan sebagai berikut

$$\frac{ds}{d\mathbf{A}} = \begin{bmatrix} \frac{ds}{da_{11}} & \frac{ds}{da_{12}} & \dots \\ \frac{ds}{da_{21}} & \frac{ds}{da_{22}} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} \quad (2.16)$$

Selanjutnya, bentuk umum  $\mathbf{P}_k$  dari persamaan 2.13 dikembangkan dan ditulis menjadi

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_k^- - \mathbf{P}_k^- \mathbf{H}_k^T \mathbf{K}_k^T + \mathbf{K}_k (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k) \mathbf{K}_k^T \quad (2.17)$$

Perlu diketahui bahwa  $\mathbf{K}_k$  pada bagian kedua dan ketiga adalah linier, sedangkan pada bagian keempat dari persamaan 2.17 adalah kuadrat. Kedua formula diferensiasi matriks (persamaan 2.14 dan 2.15) digunakan ke persamaan 2.17 guna meminimisasi trace dari  $\mathbf{P}_k$ , dimana *mean-square error* untuk setiap elemennya dapat diminimasi saat total dari *mean-square error* telah diminimisasi. Selanjutnya *trace* dari matriks  $\mathbf{P}_k$  diturunkan terhadap  $\mathbf{K}_k$ , perlu dicatat bahwa *trace* dari  $\mathbf{P}_k^- \mathbf{H}_k^T \mathbf{K}_k^T$  adalah sama dengan *trace* dari *transpose*-nya, yaitu  $\mathbf{K}_k \mathbf{H}_k \mathbf{P}_k^-$ , menghasilkan

$$\frac{d(\text{trace } \mathbf{P}_k)}{d\mathbf{K}_k} = -2(\mathbf{H}_k \mathbf{P}_k^-)^T + 2\mathbf{K}_k (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k) \quad (2.18)$$

Lalu dengan menetapkan ruas kiri (turunan) adalah sama dengan nol, penyelesaian untuk gain optimal  $\mathbf{K}_k$  adalah

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \quad (2.19)$$

Gain  $\mathbf{K}_k$  ini, adalah *gain* yang meminimisasi *mean-square estimation error*, disebut dengan *Kalman gain*.

Matriks kovarian yang berhubungan dengan estimasi optimal dapat dihitung. Merujuk pada persamaan 2.13, didapat

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T \quad (2.20)$$

$$= \mathbf{P}_k^- - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_k^- - \mathbf{P}_k^- \mathbf{H}_k^T \mathbf{K}_k^T + \mathbf{K}_k (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k) \mathbf{K}_k^T \quad (2.21)$$

Dengan substitusi persamaan 2.19 ke persamaan 2.21 didapat

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \mathbf{H}_k \mathbf{P}_k^- \quad (2.22)$$

Atau

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k) \mathbf{K}_k^T \quad (2.23)$$

Atau

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- \quad (2.24)$$

Didapat empat persamaan yang mengekspresikan perhitungan  $\mathbf{P}_k$  terbaru dari nilai  $\mathbf{P}_k^-$  sebelumnya. Tiga diantaranya, yakni persamaan 2.22, 2.23, dan 2.24, hanya valid untuk kondisi gain  $\mathbf{K}_k$  yang optimal. Sementara itu, persamaan 2.20 valid/dapat digunakan untuk nilai  $\mathbf{K}_k$  manapun, optimal ataupun sub optimal. Untuk tugas akhir ini, persamaan pembaruan (update) dari matriks *error* kovarian yang digunakan adalah yang paling sederhana, yaitu persamaan 2.24.

Sekarang data pengukuran saat  $t_k$  dapat diasimilasikan menggunakan persamaan 2.10 dengan  $\mathbf{K}_k$  adalah *Kalman gain* berdasarkan persamaan 2.19. Perlu diperhatikan bahwa diperlukan nilai  $\hat{\mathbf{x}}_k^-$  dan  $\mathbf{P}_k^-$ , untuk mendapatkan nilai tersebut. Estimasi terbaru (*updated*),  $\hat{\mathbf{x}}_k$  dapat didapatkan melalui matriks transisi. Karena  $\mathbf{w}_k$  adalah *zero mean* dan tidak berkorelasi dengan vektor-vektor  $\mathbf{w}_k$  yang lalu, maka  $\mathbf{w}_k$  dapat diabaikan, sehingga

$$\hat{\mathbf{x}}_{k+1}^- = \boldsymbol{\phi}_k \hat{\mathbf{x}}_k \quad (2.25)$$

Matriks *error* kovarian yang berhubungan dengan  $\mathbf{x}_{k+1}^-$  didapatkan dengan pertama membentuk persamaan yang mengekspresikan *error* sebelumnya

$$\begin{aligned} \mathbf{e}_{k+1}^- &= \mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1}^- \\ &= (\boldsymbol{\phi}_k \mathbf{x}_k + \mathbf{w}_k) - \boldsymbol{\phi}_k \hat{\mathbf{x}}_k \\ &= \boldsymbol{\phi}_k \mathbf{e}_k + \mathbf{w}_k \end{aligned} \quad (2.26)$$

$\mathbf{w}_k$  dan  $\mathbf{e}_k$  memiliki *zero crosscorelation*, dikarenakan  $\mathbf{w}_k$  adalah *noise* proses untuk langkah selanjutnya dari  $t_k$ . Sehingga, persamaan untuk  $\mathbf{P}_{k+1}^-$  adalah

$$\begin{aligned}\mathbf{P}_{k+1}^- &= E[\mathbf{e}_{k+1}^- \mathbf{e}_{k+1}^{-T}] = E[(\boldsymbol{\phi}_k \mathbf{e}_k + \mathbf{w}_k)(\boldsymbol{\phi}_k \mathbf{e}_k + \mathbf{w}_k)^T] \\ &= \boldsymbol{\phi}_k \mathbf{P}_k \boldsymbol{\phi}_k^T + \mathbf{Q}_k\end{aligned}\quad (2.27)$$

Persamaan 2.10, 2.19, 2.24, 2.25, dan 2.27 merupakan pembentuk persamaan rekursif Kalman filter. Pada saat simulasi, *loop* akan berhenti ketika simulasi dihentikan atau ketika data terakhir telah diproses, namun dalam praktisnya, *loop* Kalman filter dihentikan dengan *software* atau perintah pada INS.

proses perhitungan sekuensial dari Kalman filter dengan mengikuti algoritma berikut,

0. Inisialisasi, saat  $k = 0$ , nilai dari state estimasi  $\hat{\mathbf{x}}_0^-$  dan matriks kovarian *error*  $\mathbf{P}_0^-$  adalah 0.
1. Hitung nilai Kalman *gain*  $\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1}$ .
2. Update estimasi dengan nilai pengukuran  $\mathbf{z}_k$  :  $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-)$ .
3. Hitung matriks kovarian *error* untuk estimasi yang telah *update* :  $\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^-$
4. Menyelesaikan perhitungan untuk prediksi ke  $k + 1$  :  $\hat{\mathbf{x}}_{k+1}^- = \boldsymbol{\phi}_k \hat{\mathbf{x}}_k$ ,  $\mathbf{P}_{k+1}^- = \boldsymbol{\phi}_k \mathbf{P}_k \boldsymbol{\phi}_k^T + \mathbf{Q}_k$
5. Untuk  $k = k + 1$ , kembali ke proses nomor 1.

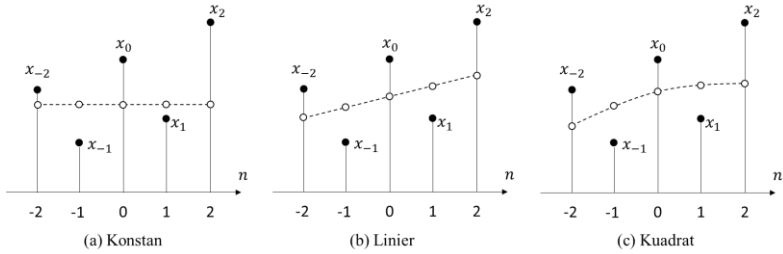
## 2.4 Algoritma *Smoothing* Savitzky-Golay

Savitzky-Golay *smoothing* filter atau yang disebut juga sebagai *polynomial smoothing*, atau filter *least-square smoothing*, adalah generalisasi dari filter FIR yang mampu menjaga sinyal frekuensi tinggi dari sinyal yang diinginkan, namun tidak menghapus *noise* sebanyak filter FIR sebagai gantinya. Dikarenakan ada batasan dalam aplikasi filter *averaging* FIR. Untuk mencapai reduksi *noise* orde tinggi, nilai length  $N$  diperlukan sangat besar, sehingga frekuensi *cut-off* dari filter FIR  $\omega_c = \pi/N$  menjadi lebih kecil, hal ini akan menjadikan sinyal asli yang berada pada frekuensi lebih tinggi ikut terfilter [3].

Filter *smoothing* bekerja secara optimal mencocokkan satu set data poin ke dalam derajat polinomial yang berbeda. Gambar 2.10 menunjukkan adanya lima sampel sinyal *bernoise* ( $x_{-2}, x_{-1}, x_0, x_1, x_2$ )



yang terletak simetris pada data ke-0 yang kemudian akan bergeser ke kanan searah berjalannya waktu.



**Gambar 2.9** *Smoothing* data dengan polinomial orde  $d = 0, 1, 2$ .

Pada gambar 2.10, dilakukan *fitting* lima data ke sinyal konstan, linier dan kuadratik. Nilai yang telah diperhalus (*smoothed*) pada masing-masing sinyal adalah seperti yang diberikan pada polinomial ke-0, ke-1 dan ke-2 untuk  $m = -2, -1, 0, 1, 2$ :

$$x_m = c_0 \text{ (konstan)}$$

$$x_m = c_0 + c_1 m \text{ (linier)} \quad (2.28)$$

$$x_m = c_0 + c_1 m + c_2 m^2 \text{ (kuadratik)}$$

Untuk setiap orde polinomial yang dipilih, koefisien  $c_i$  harus ditentukan dengan optimal agar kurva polinomial dari orde yang bersangkutan dapat sejajar dengan data. Hal ini dapat dicapai dengan menggunakan *least-square fit*, yaitu dengan memilih koefisien  $c_i$  yang meminimasi total *mean-square error*. Pada kasus kuadratik, diketahui indeks performansi yang akan diminimasi sebagai berikut:

$$J = \sum_{m=-2}^2 e_m^2 = \sum_{m=-2}^2 (x_m - (c_0 + c_1 m + c_2 m^2))^2 = \min \quad (2.29)$$

Dimana *error fitting* didefinisikan sebagai berikut

$$e_m = x_m - \hat{x}_m = x_m - (c_0 + c_1 m + c_2 m^2), \quad m = -2, -1, 0, 1, 2$$

Untuk menyederhanakan penulisan dari persamaan 2.28 dan 2.29, yang nantinya bisa digunakan untuk orde yang lebih tinggi dan jumlah data yang lebih banyak, maka didefinisikan vektor dimensi lima sebagai berikut:

$$\mathbf{x} = [x_{-2} \quad x_{-1} \quad x_0 \quad x_1 \quad x_2]^T$$

$$\hat{\mathbf{x}} = [\hat{x}_{-2} \quad \hat{x}_{-1} \quad \hat{x}_0 \quad \hat{x}_1 \quad \hat{x}_2]^T$$

$$\mathbf{e} = [e_{-2} \quad e_{-1} \quad e_0 \quad e_1 \quad e_2]^T = \mathbf{x} - \hat{\mathbf{x}}$$

Dengan cara yang sama, didefinisikan vektor basis polinomial dimensi lima, yakni  $s_0, s_1, s_2$  yang komponennya adalah sebagai berikut:

$$s_0(m) = 1, s_1(m) = m, s_2(m) = m^2, \quad -2 \leq m \leq 2$$

Yang secara vektor dapat ditulis:

$$\begin{aligned} \mathbf{s}_0 &= [1 \ 1 \ 1 \ 1 \ 1]^T \\ \mathbf{s}_1 &= [-2 \ -1 \ 0 \ 1 \ 2]^T \\ \mathbf{s}_2 &= [4 \ 1 \ 0 \ 1 \ 4]^T \end{aligned} \quad (2.30)$$

Dengan demikian persamaan 2.28 dapat ditulis secara vektor sebagai:

$$\hat{\mathbf{x}} = c_0 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + c_1 \begin{bmatrix} -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{bmatrix} + c_2 \begin{bmatrix} 4 \\ 1 \\ 0 \\ 1 \\ 4 \end{bmatrix} = c_0 \mathbf{s}_0 + c_1 \mathbf{s}_1 + c_2 \mathbf{s}_2$$

Sehingga

$$\hat{\mathbf{x}} = c_0 \mathbf{s}_0 + c_1 \mathbf{s}_1 + c_2 \mathbf{s}_2 = [\mathbf{s}_0 \ \mathbf{s}_1 \ \mathbf{s}_2] \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \mathbf{S} \mathbf{c} \quad (2.31)$$

Dengan matriks basis  $\mathbf{S}$  (5x3) adalah sebagai berikut

$$\mathbf{S} = [\mathbf{s}_0 \ \mathbf{s}_1 \ \mathbf{s}_2] = \begin{bmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \quad (2.32)$$

Dengan persamaan  $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}} = \mathbf{x} - \mathbf{S} \mathbf{c}$ , indeks performansi pada persamaan 2.29 dapat ditulis sebagai *dot product* sebagai berikut

$$\mathcal{J} = \mathbf{e}^T \mathbf{e} = (\mathbf{x} - \mathbf{S} \mathbf{c})^T (\mathbf{x} - \mathbf{S} \mathbf{c}) = \mathbf{x}^T \mathbf{x} - 2 \mathbf{c}^T \mathbf{S}^T \mathbf{x} + \mathbf{c}^T \mathbf{S}^T \mathbf{S} \mathbf{c} \quad (2.33)$$

Untuk meminimasi persamaan tersebut terhadap  $\mathbf{c}$ , gradien dari  $\frac{\partial \mathcal{J}}{\partial \mathbf{c}}$  menjadi sama dengan nol sebagai berikut:

$$\frac{\partial \mathcal{J}}{\partial \mathbf{c}} = -2 \mathbf{S}^T \mathbf{e} = -2 \mathbf{S}^T (\mathbf{x} - \mathbf{S} \mathbf{c}) = -2 (\mathbf{S}^T \mathbf{x} - \mathbf{S}^T \mathbf{S} \mathbf{c}) \quad (2.34)$$

Syarat optimasi:

$$\frac{\partial \mathcal{J}}{\partial \mathbf{c}} = 0 \Rightarrow \mathbf{S}^T \mathbf{e} = 0 \quad (2.35)$$

Yang dapat ditulis menjadi

$$\mathbf{S}^T \mathbf{S} \mathbf{c} = \mathbf{S}^T \mathbf{x} \quad (2.36)$$

Sehingga solusi optimal untuk  $\mathbf{c}$  adalah

$$\mathbf{c} = (\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T \mathbf{x} = \mathbf{G}^T \mathbf{x} \quad (2.37)$$

Dimana matriks  $\mathbf{G}$  adalah

$$\mathbf{G} = \mathbf{S}(\mathbf{S}^T \mathbf{S})^{-1} \quad (2.38)$$

Dengan memasukkan optimal koefisien  $\mathbf{c}$  ke persamaan 2.31, didapat nilai yang sudah dihaluskan (*smoothed*):

$$\hat{\mathbf{x}} = \mathbf{S}\mathbf{c} = \mathbf{S}\mathbf{G}^T \mathbf{x} = \mathbf{S}(\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T \mathbf{x} \equiv \mathbf{B}\mathbf{x} \quad (2.39)$$

Dimana matriks  $\mathbf{B}$  (5x5) didefinisikan dengan

$$\mathbf{B} = \mathbf{S}\mathbf{G}^T = \mathbf{G}\mathbf{S}^T = \mathbf{S}(\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T \quad (2.40)$$

Kemudian didefinisikan matriks simetris 3x3, yakni  $\mathbf{F} = \mathbf{S}^T \mathbf{S}$ , yang mana muncul pada persamaan untuk matriks  $\mathbf{G}$  dan  $\mathbf{B}$ . Menggunakan persamaan 2.32 didapat

$$\mathbf{F} = \mathbf{S}^T \mathbf{S} = \begin{bmatrix} \mathbf{s}_0^T \\ \mathbf{s}_1^T \\ \mathbf{s}_2^T \end{bmatrix} [\mathbf{s}_0 \quad \mathbf{s}_1 \quad \mathbf{s}_2] = \begin{bmatrix} \mathbf{s}_0^T \mathbf{s}_0 & \mathbf{s}_0^T \mathbf{s}_1 & \mathbf{s}_0^T \mathbf{s}_2 \\ \mathbf{s}_1^T \mathbf{s}_0 & \mathbf{s}_1^T \mathbf{s}_1 & \mathbf{s}_1^T \mathbf{s}_2 \\ \mathbf{s}_2^T \mathbf{s}_0 & \mathbf{s}_2^T \mathbf{s}_1 & \mathbf{s}_2^T \mathbf{s}_2 \end{bmatrix} \quad (2.41)$$

Menggunakan persamaan 2.32, nilai  $\mathbf{F}$  dan  $\mathbf{F}^{-1}$  adalah sebagai berikut

$$\mathbf{F} = \begin{bmatrix} 5 & 0 & 10 \\ 0 & 10 & 0 \\ 10 & 0 & 34 \end{bmatrix}, \quad \mathbf{F}^{-1} = \frac{1}{35} \begin{bmatrix} 17 & 0 & -5 \\ 0 & 3.5 & 0 \\ -5 & 0 & 2.5 \end{bmatrix} \quad (2.42)$$

Kemudian, menghitung matriks  $\mathbf{G}$

$$\mathbf{G} = \mathbf{S}\mathbf{F}^{-1} = \frac{1}{35} \begin{bmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \begin{bmatrix} 17 & 0 & -5 \\ 0 & 3.5 & 0 \\ -5 & 0 & 2.5 \end{bmatrix} \text{ atau}$$

$$\mathbf{G} = \frac{1}{35} \begin{bmatrix} -3 & -7 & 5 \\ 12 & -3.5 & -2.5 \\ 17 & 0 & -5 \\ 12 & 3.5 & -2.5 \\ -3 & 7 & 5 \end{bmatrix} \equiv [\mathbf{g}_0 \quad \mathbf{g}_1 \quad \mathbf{g}_2] \quad (2.43)$$

Selanjutnya, dengan menggunakan persamaan 2.40, matriks  $\mathbf{B}$  (5x5) dapat dihitung

$$\mathbf{B} = \mathbf{G}\mathbf{S}^T = \frac{1}{35} \begin{bmatrix} -3 & -7 & 5 \\ 12 & -3.5 & -2.5 \\ 17 & 0 & -5 \\ 12 & 3.5 & -2.5 \\ -3 & 7 & 5 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \\ 4 & 1 & 0 & 1 & 4 \end{bmatrix} \text{ atau}$$

$$\mathbf{B} = \frac{1}{35} \begin{bmatrix} 31 & 9 & -3 & -5 & 3 \\ 9 & 13 & 12 & 6 & -5 \\ -3 & 12 & 17 & 12 & -3 \\ -5 & 6 & 12 & 13 & 9 \\ 3 & -5 & -3 & 9 & 31 \end{bmatrix} \equiv [\mathbf{b}_{-2} \quad \mathbf{b}_{-1} \quad \mathbf{b}_0 \quad \mathbf{b}_1 \quad \mathbf{b}_2] \quad (2.44)$$

Dikarenakan matriks  $\mathbf{B}$  merupakan matriks simetris, maka elemen barisnya sama dengan elemen kolom, sehingga dapat dituliskan dalam bentuk baris ataupun kolom sebagai berikut:

$$\mathbf{B} = [\mathbf{b}_{-2} \quad \mathbf{b}_{-1} \quad \mathbf{b}_0 \quad \mathbf{b}_1 \quad \mathbf{b}_2] = \begin{bmatrix} \mathbf{b}_{-2}^T \\ \mathbf{b}_{-1}^T \\ \mathbf{b}_0^T \\ \mathbf{b}_1^T \\ \mathbf{b}_2^T \end{bmatrix} = \mathbf{B}^T$$

Kelima kolom atau baris dari matriks  $\mathbf{B}$  merupakan Savitzky-Golay *smoothing* filter dengan lebar 5 dan orde polinomial 2. Nilai ter-*smoothing*  $\hat{x}$  dapat dituliskan sebagai berikut:

$$\begin{bmatrix} \hat{x}_{-2} \\ \hat{x}_{-1} \\ \hat{x}_0 \\ \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} = \hat{\mathbf{x}} = \mathbf{B}\mathbf{x} = \mathbf{B}^T\mathbf{x} = \begin{bmatrix} \mathbf{b}_{-2}^T \\ \mathbf{b}_{-1}^T \\ \mathbf{b}_0^T \\ \mathbf{b}_1^T \\ \mathbf{b}_2^T \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{b}_{-2}^T\mathbf{x} \\ \mathbf{b}_{-1}^T\mathbf{x} \\ \mathbf{b}_0^T\mathbf{x} \\ \mathbf{b}_1^T\mathbf{x} \\ \mathbf{b}_2^T\mathbf{x} \end{bmatrix}$$

Atau, untuk  $m = -2, -1, 0, 1, 2$

$$\hat{x}_m = \mathbf{b}_m^T\mathbf{x} \quad (2.43)$$

Sehingga vektor filter ke- $m$   $\mathbf{b}_m$  dikalikan dengan vektor data  $\mathbf{x}$ , akan menghasilkan nilai ter-*smoothing* ke- $m$  dari data sampel.

Dari kelima kolom matriks  $\mathbf{B}$ , kolom tengah,  $\mathbf{b}_0$  adalah yang paling penting peranannya karena  $\mathbf{b}_0$  memfiltrasi data  $x_0$ , yang terletak secara simetris terhadap data sampling lainnya seperti pada gambar . Dalam melakukan *smoothing* terhadap blok data, vektor  $\mathbf{b}_0$  digunakan pada saat keadaan *steady-state*, yang mana kolom lain dari matriks  $\mathbf{B}$  digunakan hanya saat *input-on* dan *input-off* transien.

Desain Savitzky-Golay filter (S-G filter) dapat diturunkan secara langsung dari pendekatan polinomial orde  $d$  dengan lebar data  $N$  dari vektor data  $\mathbf{x}$  adalah sebagai berikut:

Asumsikan lebar data  $N$  adalah ganjil, misalkan  $N = 2M + 1$ , vektor data  $\mathbf{x}$  dapat dituliskan sebagai berikut:

$$\mathbf{x} = [x_{-M} \quad \dots \quad x_{-1} \quad x_0 \quad x_1 \quad \dots \quad x_M]^T \quad (2.44)$$

Kemudian dilakukan fitting polinomial orde  $d$  oleh data sampling  $x$  dengan lebar  $N$ , generalisasi dari persamaan 2.28 didapat

$$\hat{x}_m = c_0 + c_1 m + \dots + c_d m^d, \quad -M \leq m \leq M \quad (2.45)$$

Pada kasus ini, terdapat  $d + 1$  vektor basis polinomial  $s_i = 0, 1, 2, \dots, d$ , didefinisikan memiliki komponen sebagai berikut

$$s_i(m) = m^i, \quad -M \leq m \leq M \quad (2.46)$$

Sedangkan matriks  $S$  berdimensi  $N \times (d + 1)$  dengan  $s_i$  sebagai kolom didefinisikan sebagai berikut

$$\mathbf{S} = [\mathbf{s}_0 \quad \mathbf{s}_1 \quad \dots \quad \mathbf{s}_d] \quad (2.47)$$

Nilai hasil *smoothing* dari persamaan 2.45 dapat ditulis dalam bentuk vektor:

$$\hat{\mathbf{x}} = \sum_{i=0}^d c_i \mathbf{S}_i = [\mathbf{s}_0 \quad \mathbf{s}_1 \quad \dots \quad \mathbf{s}_d] \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_d \end{bmatrix} = \mathbf{S} \mathbf{c} \quad (2.48)$$

Langkah-langkah mendesain Savitzky-Golay *smoothing* filter adalah sebagai berikut:

$$\mathbf{F} = \mathbf{S}^T \mathbf{S} \Leftrightarrow \mathbf{F}_{ij} = \mathbf{s}_i^T \mathbf{s}_j, \quad i, j = 0, 1, \dots, d$$

$$\mathbf{G} = \mathbf{S} \mathbf{F}^{-1} \equiv [\mathbf{g}_0 \quad \mathbf{g}_1 \quad \dots \quad \mathbf{g}_d] \quad (2.49)$$

$$\mathbf{B} = \mathbf{S} \mathbf{G}^T = \mathbf{G} \mathbf{S}^T = \mathbf{S} \mathbf{F}^{-1} \mathbf{S}^T \equiv [\mathbf{b}_{-M} \quad \dots \quad \mathbf{b}_0 \quad \dots \quad \mathbf{b}_M]$$

Vektor koefisien  $\mathbf{c}$  dan vektor data ter-*smoothing* adalah:

$$\mathbf{c} = \mathbf{G}^T \mathbf{x} \Leftrightarrow c_i = \mathbf{g}_i^T \mathbf{x}, \quad i = 0, 1, \dots, d$$

$$\hat{\mathbf{x}} = \mathbf{B} \mathbf{x} \Leftrightarrow \hat{x}_m = \mathbf{b}_m^T \mathbf{x}, \quad -M \leq m \leq M \quad (2.50)$$

Nilai tengah dari hasil ter-*smoothing*  $y_0 = \hat{x}_0$  pada filter SG  $\mathbf{b}_0$ :

$$y_0 = \mathbf{b}_0^T \mathbf{x} = \sum_{m=-M}^M b_0(m) x_m \quad (2.51)$$

Vektor data  $x$  berdimensi  $N$  bergeser menuju waktu ke- $n$  dengan:

$$\mathbf{x} \rightarrow [x_{n-M} \quad \dots \quad x_{n-1} \quad x_n \quad x_{n+1} \quad \dots \quad x_{n+M}]^T$$

Sehingga filter Savitzky-Golay dengan lebar  $N$  dan orde  $d$  untuk *smoothing* data bernoise  $x(n)$  dalam keadaan steady-state adalah:

$$y(n) = \sum_{m=-M}^M \mathbf{b}_0(m) x(n+m) \quad (2.52)$$

Dimana  $y(n)$  merupakan nilai keluaran filter,  $\mathbf{b}_0$  merupakan vektor parameter pendekatan polinomial orde ke- $d$  (biasanya menggunakan orde 2 atau lebih), dan  $x(n+m)$  merupakan sinyal masukan ke- $(n+m)$ .

## 2.5 Model *Error* INS

Model *error* INS adalah sebuah model matematika dari *error* yang dihasilkan INS, atau lebih tepatnya disebut juga model dari estimasi *noise*. Untuk mendapatkan model ini, pertama kita harus mendapatkan selisih dari sinyal IMU (percepatan dan keceptan sudut) dengan sinyal sebenarnya. Namun karena model INS hanya menggunakan data dari INS, sulit untuk mendapatkan sinyal sebenarnya tanpa adanya bantuan sistem navigasi lain seperti GPS. Oleh karena itu, untuk mendapatkan nilai estimasi awal *noise*, digunakan filter LPF dan HPF.

Persamaan model *error* dari INS adalah sebagai berikut:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/R_e & 0 & 0 & 0 & 0 & 0 & 0 & \omega_x \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/R_e & 0 & 0 & 0 & \omega_y \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} + \begin{bmatrix} 0 \\ u_{ax} \\ u_{gx} \\ 0 \\ u_{ay} \\ u_{gy} \\ 0 \\ u_{az} \\ u_{gz} \end{bmatrix} \quad (2.53)$$

Dimana

$x_1$  = *error* posisi arah east (sumbu-x) [m]

$x_2$  = *error* kecepatan arah east (sumbu-x) [m/s]

$x_3$  = kemiringan benda terhadap sumbu-y [rad]

$x_4$  = *error* posisi arah north (sumbu-y) [m]

$x_5$  = *error* kecepatan arah north (sumbu-y) [m/s]

$x_6$  = kemiringan benda terhadap sumbu-(-x) [rad]

$x_7$  = *error* posisi vertikal [m]

$x_8$  = *error* kecepatan vertikal [m/s]

$x_9$  = *error* sudut azimuth benda [rad]

$u_{ax}$  = input *error accelerometer* sumbu-x

$u_{gx}$  = input *error gyroscope* sumbu-x

$u_{ay}$  = input *error accelerometer* sumbu-y

$u_{gy}$  = input *error gyroscope* sumbu-y

$u_{az}$  = input *error accelerometer* sumbu-z

$u_{gz}$  = input *error gyroscope* sumbu-z

Dengan

$g$  = percepatan gravitasi [ $9.8 \text{ m/s}^2$ ]

$R_e$  = jari-jari bumi pada ekuator [ $6,380,000 \text{ m}$ ]

$\omega_y$  = kecepatan sudut pada sumbu-y [ $7.292115 * 10^{-5} \text{ rad/s}$ ]

$\omega_x$  = kecepatan sudut pada sumbu-x [ $-1.57 * 10^{-5} \text{ rad/s}$ ]

Untuk mendapatkan nilai dari anggota vektor  $U$  yang merupakan input/masukan berupa *error* dari *Accelerometer* dan *Gyroscope*, digunakan filter LPF dan HPF. Data percepatan ber-*noise* difilter oleh LPF, menghasilkan data percepatan dengan *noise* teredam, sementara untuk data kecepatan sudut ber-*noise*, difilter dengan HPF menghasilkan data kecepatan sudut dengan *noise* teredam. Kemudian nilai sebelum filter diselisihkan dengan nilai setelah filtrasi, baik untuk LPF dan HPF menghasilkan *error accelerometer* dan *gyroscope*.

Metode filtrasi untuk mendapatkan model dari *noise* disini adalah dengan menggunakan metode Butterworth. Filter Butterworth adalah filter yang memiliki magnitudo paling datar dan tidak mengandung *ripple*. Metode filtrasi ini dikenalkan oleh seorang insinyur dan fisikiawan Inggris, Stephen Butterworth pada tahun 1930 dalam papernya yang berjudul “*On the Theory of Filter Amplifier*”.

Nilai *gain*  $G(\omega)$  dari orde ke- $n$  filter Butterworth dirumuskan sebagai berikut:

$$G^2(\omega) = |H(j\omega)|^2 = \frac{G_0^2}{1 + \left(\frac{j\omega}{j\omega_c}\right)^{2n}} \quad (2.54)$$

Dimana

$n$  adalah orde dari filter Butterworth

$\omega_c$  adalah frekuensi *cut-off*

$G_0$  adalah gain filter saat frekuensi bernilai nol

Kemudian, dengan menggunakan transformasi Laplace, didapat

$$H(s) = \frac{G_0}{\prod_{k=1}^n (s - s_k) / \omega_c} \quad (2.55)$$

Dengan

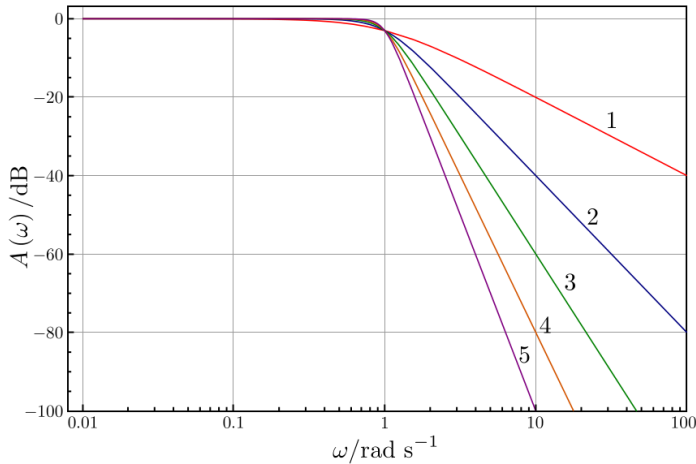
$$s_k = \omega_c e^{\frac{j(2k-1)\pi}{n}} \quad k = 1, 2, 3, \dots, n$$

Persamaan polinomial Butterworth yang kompleks dapat dinormalisasikan dengan men-setting  $\omega_c = 1$ . Tabel 2.2 adalah tabel berisi normalisasi polinomial Butterworth berdasarkan ordenya.

**Tabel 2.2** Tabel normalisasi polinomial Butterworth

$n$	Faktor Polinomial $B_n(s)$
1	$(s + 1)$
2	$(s^2 + 1.4142s + 1)$
3	$(s + 1)(s^2 + s + 1)$
4	$(s^2 + 0.7654s + 1)(s^2 + 1.8478s + 1)$
5	$(s + 1)(s^2 + 0.6180s + 1)(s^2 + 1.6180s + 1)$
6	$(s^2 + 0.5176s + 1)(s^2 + 1.4142s + 1)(s^2 + 1.9319s + 1)$
7	$(s + 1)(s^2 + 0.4450s + 1)(s^2 + 1.2470s + 1)(s^2 + 1.8019s + 1)$
8	$(s^2 + 0.3902s + 1)(s^2 + 1.1111s + 1)(s^2 + 1.6629s + 1)(s^2 + 1.9616s + 1)$

Sedangkan gambar 2.10 adalah gambar plot dari *gain* Butterworth *low-pass filter* untuk orde ke-1 sampai ke-5 dengan frekuensi *cut-off*  $\omega_c = 1$ .



**Gambar 2.10** *Gain* Butterworth untuk orde 1 sampai 5

## 2.6 Sistem Navigasi Terintegrasi

INS adalah sistem navigasi otonom yang menyediakan data dengan *bandwith* yang mampu melebihi 100 Hz, sistem navigasi ini memiliki akurasi jangka pendek yang baik, dan mempunyai informasi tambahan berupa orientasi dari kendaraan atau benda bergerak selain kecepatan dan posisi. Namun *error* jangka panjang cenderung bertambah tanpa batas seraya *error* dari INS berakumulasi dikarenakan algoritma integral pada IMU. Sebaliknya, GPS memiliki *error* jangka panjang yang

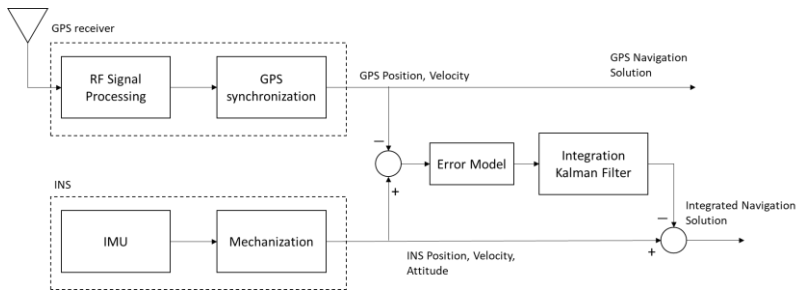


kecil dan terbatas pada beberapa meter saja. Namun GPS memiliki output *data rate* yang kecil dan tidak memiliki informasi orientasi (hanya posisi dan kecepatan). GPS *receiver* memerlukan setidaknya empat satelit yang terbaca secara langsung, hal ini menyebabkan sistem navigasi berbasis satelit ini sering mengalami gangguan dari gedung tinggi, pohon, dan terowongan.

Karakteristik yang saling berkomplemen ini menghasilkan sebuah sistem navigasi terintegrasi yang mampu menutupi kelemahan masing-masing dan menghasilkan data keluaran dengan keakuratan yang lebih baik. *Estimator* (pada kasus ini menggunakan Kalman filter) diletakkan setelah data antara INS dan GPS diselisihkan, menghasilkan sebuah *error* model dari sistem terintegrasi antara INS dan GPS. Ada 3 macam sistem navigasi terintegrasi antara INS dan GPS, yakni sistem terintegrasi *loosely-coupled*, *tightly-coupled* dan *ultra-tight*.

### **2.6.1 Sistem Navigasi Terintegrasi *Loosely-Coupled***

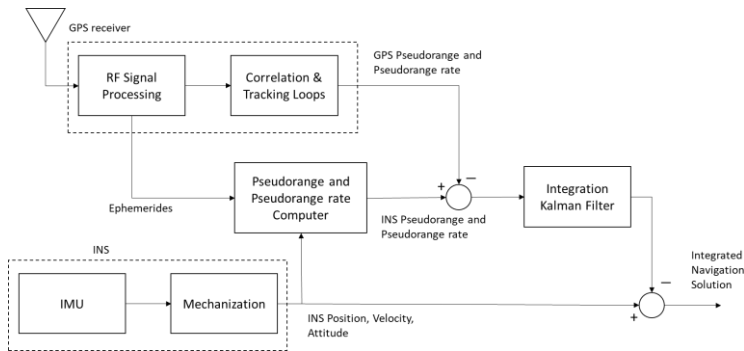
Pada konfigurasi integrasi ini, INS dan GPS bekerja secara sendiri-sendiri dan menyediakan solusi navigasi sendiri-sendiri. Solusi navigasi dari masing-masing sistem navigasi ini diselisihkan untuk menghasilkan model *error*. Sebelum solusi navigasi diselisihkan, data keluaran GPS disinkronisasi agar memiliki *data rate* yang sama dengan INS. Setelahnya, kalman filter mengestimasi *error* yang didapat dari model *error* sistem terintegrasi. Gambar 2.11 merupakan blok diagram dari sistem navigasi terintegrasi *loosely-coupled*. Kelebihan sistem terintegrasi ini adalah mudah diimplementasikan dan *robust*, namun tidak ada koreksi untuk data dari GPS, terutama dari gangguan seperti kurangnya satelit yang terbaca oleh *receiver*, gedung-gedung tinggi, pohon, terowongan dan degradasi saat sinyal melewati atmosfer dan lain-lain. Oleh karenanya, untuk mensimulasikan model sistem terintegrasi ini, GPS diasumsikan sebagai penyedia data navigasi tanpa *noise*.



**Gambar 2.11** Sistem Navigasi Terintegrasi *Loosely-Coupled*

### 2.6.2 Sistem Navigasi Terintegrasi *Tightly-Coupled*

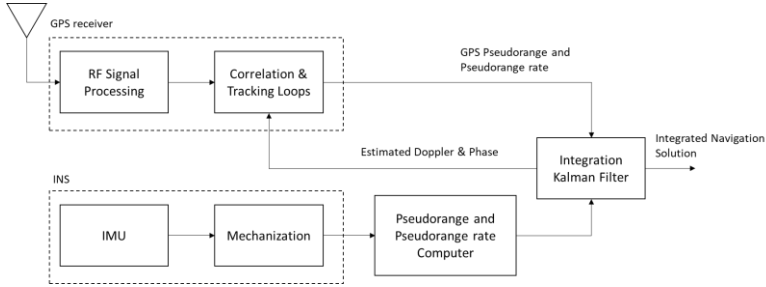
Pada tipe integrasi ini, selisih dari pseudo-range dan pseudo-range rate dari GPS diprediksi oleh INS dan dimasukkan ke Kalman filter untuk mengestimasi *error* pada INS. Keluaran dari INS kemudian dikoreksi dengan *error* yang diestimasi oleh Kalman filter. Diagram blok dari sistem navigasi terintegrasi ini ditunjukkan oleh gambar 2.12. Integrasi jenis ini mampu mengatasi masalah yang terjadi pada sistem terintegrasi *loosely-coupled*, yaitu adanya data pengukuran yang berkolerasi, juga mampu memberikan pembaruan (*update*) data GPS bahkan ketika satelit yang tertangkap oleh *receiver* kurang dari empat buah. Hal ini sangat membantu dalam aplikasi pada kehidupan sehari-hari dikarenakan lintasan yang sering dilalui adalah lintasan pada lingkungan perkotaan dengan gedung tinggi, daerah hutan dan medan yang tidak rata yang menjadi salah satu gangguan pada sistem navigasi satelit. Sebaliknya, *tightly-coupled* lebih rumit untuk diimplementasikan karena adanya algoritma yang melibatkan pemrosesan data mentah GPS, dan tidak adanya solusi navigasi GPS (solusi navigasi GPS tidak dihitung/diproses karena data mentah GPS langsung digunakan untuk perhitungan *pseudo-range* dan/atau *pseudo-range rate*).



**Gambar 2.12** Sistem Navigasi Terintegrasi *Tightly-Coupled*

### 2.6.3 Sistem Navigasi Terintegrasi *Ultra-Tight*

Sistem terintegrasi tipe ini disebut juga deep integration, dan ditunjukkan pada gambar 2.13. Ada dua perbedaan besar antara sistem terintegrasi *ultra-tight* dengan dua jenis sistem terintegrasi lain. Pertama, terdapat perbedaan susunan dari *receiver* GPS untuk menyediakan implementasi yang berbeda pada *tracking loop* untuk estimasi posisi. Kedua, informasi dari INS digunakan sebagai bagian integral pada perhitungan yang terjadi di *receiver* GPS, hal ini menyebabkan INS dan GPS bukan lagi sistem navigasi yang independen satu sama lain. Tipe integrasi jenis ini juga menambah kompleksitas pada implementasinya, karena diharuskan untuk mengubah internal perangkat keras GPS *receiver*. Kelebihan dari sistem terintegrasi ini adalah tersedianya sebuah data GPS yang bebas dari gangguan, terutama *jamming*, dan bekerja pada SNR (*Signal to Noise Ratio*) yang rendah serta mampu menghasilkan solusi navigasi meskipun jumlah satelit yang tertangkap/terbaca oleh *receiver* kurang dari empat.



**Gambar 2.13** Sistem Navigasi Terintegrasi *Ultra-Tight*

Pada tugas akhir yang dilakukan, penulis memilih menggunakan sistem navigasi terintegrasi *loosely-coupled* dikarenakan blok diagram sistem dan algoritma perhitungan yang mudah sangat cocok untuk simulasi, model sistem terintegrasi ini juga mudah untuk diimplementasikan sehingga dapat dijadikan pertimbangan untuk penelitian berikutnya mengenai sistem navigasi.

## 2.7 Model Error Sistem Terintegrasi

Pada sistem terintegrasi terutama pada sistem terintegrasi *loosely-coupled* yang digunakan pada tugas akhir ini, terdapat perubahan pada model *error* sistem. Model *error* pada sistem integrasi menjadi 15 state dengan sembilan state awal adalah sama dengan state pada model *error* INS, sedangkan untuk state ke 10 hingga 15 berisi nilai bias *accelerometer* (3 sumbu) dan bias *gyroscope* (3 sumbu). Berdasarkan [4], model *error* dari sistem terintegrasi adalah sama seperti persamaan (2.56), namun state untuk *error* bias dituliskan menjadi matriks augmented sebagai berikut:

$$\begin{bmatrix} \mathbf{x}_{1-9} \\ \mathbf{x}_{10-15} \end{bmatrix} = \begin{bmatrix} \Phi_{INS} & \mathbf{C} \\ \mathbf{0} & \Phi_{sens} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{1-9} \\ \mathbf{x}_{10-15} \end{bmatrix} + \begin{bmatrix} \mathbf{w}_{1-9} \\ \mathbf{w}_{10-15} \end{bmatrix} \quad (2.56)$$

Matriks  $\Phi_{INS}$  adalah sama dengan matriks yang ada pada persamaan (2.3), sedangkan  $\mathbf{0}$  adalah submatriks *zeros*,  $\Phi_{sens}$  adalah submatriks *state* transisi untuk bias *accelerometer* pada tiga sumbu dan bias *gyroscope* pada tiga sumbu.

$$\Phi_{sens} = \begin{bmatrix} \phi_{ax} & 0 & 0 & 0 & 0 & 0 \\ 0 & \phi_{ay} & 0 & 0 & 0 & 0 \\ 0 & 0 & \phi_{az} & 0 & 0 & 0 \\ 0 & 0 & 0 & \phi_{gx} & 0 & 0 \\ 0 & 0 & 0 & 0 & \phi_{gy} & 0 \\ 0 & 0 & 0 & 0 & 0 & \phi_{gz} \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Sedangkan untuk model pengukuran, terdapat perubahan yaitu pada statenya menggunakan *error* dari selisih posisi dan kecepatan antara INS dan GPS.

$$\begin{bmatrix} z_{pos-x} \\ z_{pos-y} \\ z_{pos-z} \\ z_{vel-x} \\ z_{vel-y} \\ z_{vel-z} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ \vdots \\ x_{10-15} \end{bmatrix} + \begin{bmatrix} v_{pos-x} \\ v_{pos-y} \\ v_{pos-z} \\ v_{vel-x} \\ v_{vel-y} \\ v_{vel-z} \end{bmatrix} \quad (2.57)$$

Untuk nilai bias dari *accelerometer* dan *gyroscope*, penulis memilih menggunakan data dari referensi [3] pada tabel 2.3 berikut,

**Tabel 2.3** Komparasi nilai bias INS

	Kualitas INS			
	Tingkat navigasi	Tingkat taktis		Tingkat Automotif
		High quality	Low quality	
Gyro bias (deg/h)	< 0.01	0.1 – 1	10	>100

Accelerometer bias ( $10^{-3} \text{ m/s}^2$ )	0.01-0.05	0.2 – 0.5	1.0 – 10.0	>10
--	-----------	-----------	------------	-----

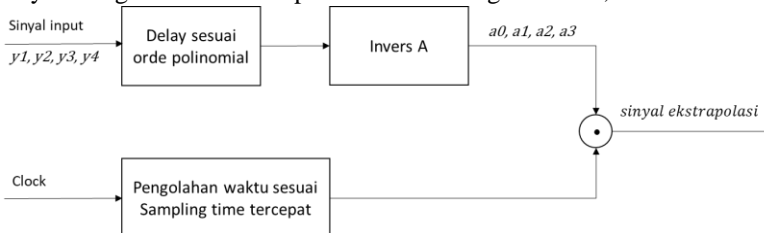
Untuk keperluan simulasi, dipilih nilai bias *gyro* adalah  $1.0 \text{ deg/h}$  atau  $4.85 * 10^{-6} \text{ rad/s}$ , sementara untuk nilai bias *accelerometer* adalah  $0.5 * 10^{-3} \text{ m/s}^2$  atau  $0.0005 \text{ m/s}^2$ .

## 2.8 Sinkronisasi data GPS

Pada aplikasinya, *data rate* INS dan GPS tidaklah sama. Sensor IMU membaca data dengan *rate* yang tinggi, menghasilkan data dengan *time sampling* yang kecil, namun mengandung banyak *noise*. Sementara *receiver* GPS mengeluarkan data dengan *rate* yang lebih rendah, menghasilkan data keluaran yang memiliki *time sampling* lebih besar, namun memiliki *error* yang sedikit dibandingkan INS karena prosesor navigasi pada modul GPS sudah melakukan kalkulasi terlebih dahulu sebelum mengeluarkan data.

Kebanyakan INS bekerja pada *rate* 50 hingga 200 Hz, sementara GPS bekerja pada *rate* 10 hingga 20 Hz. Pada tugas akhir ini, penulis memilih *data rate* INS sebesar 100 Hz dan GPS sebesar 1 Hz. Melihat bahwa adanya perbedaan pada *data rate* dari kedua sistem, perlu adanya metode sinkronisasi untuk menyamakan *data rate* keduanya.

Metode sinkronisasi yang dipilih oleh penulis untuk sistem terintegrasi adalah ekstrapolasi. Ekstrapolasi merupakan prediksi dari nilai data ke  $k+1$  dengan menggunakan pendekatan polinomial pada sinyal. Diagram blok ekstrapolasi adalah sebagai berikut,



**Gambar 2.14** Diagram blok proses ekstrapolasi

Dipilih ekstrapolasi dengan pendekatan polinomial orde 3, dengan persamaan

$$y_t = a_0 + a_1 * t + a_2 * t^2 + a_3 * t^3 \quad (2.58)$$

Untuk  $t = 1$  hingga 4, didapatkan persamaan-persamaan sebagai berikut,

$$t = 1 \rightarrow y_1 = a_0 + a_1 + a_2 + a_3$$

$$t = 2 \rightarrow y_2 = a_0 + 2a_1 + 4a_2 + 8a_3$$

$$t = 3 \rightarrow y_3 = a_0 + 3a_1 + 9a_2 + 27a_3$$

$$t = 4 \rightarrow y_4 = a_0 + 4a_1 + 16a_2 + 64a_3$$

Persamaan (2.57) sampai dengan (2.60) dapat direpresentasikan ke dalam bentuk matriks sebagai berikut

$$\mathbf{y} = \mathbf{A} * \mathbf{a} \quad (2.59)$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 4 & 16 & 64 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad (2.60)$$

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 4 & 16 & 64 \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \quad (2.61)$$

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 4 & -6 & 4 & -1 \\ -4.333 & 9.5 & -7 & 1.833 \\ 1.5 & -4 & 3.5 & -1 \\ -0.167 & 0.5 & -0.5 & 0.167 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \quad (2.62)$$

Dimana vektor  $a$  merupakan parameter polinomial orde 3. Nilai parameter ini digunakan untuk melakukan ekstrapolasi mulai pada  $t = 4.01$  hingga  $4.99$  detik. Setelah data ke  $t = 5$  dimasukkan, dilakukan perhitungan parameter polinomial kembali menggunakan data pengukuran  $y_t$  saat  $t = 2,3,4,5$  detik. Kemudian sama seperti sebelumnya, dilakukan ekstrapolasi untuk data pada  $t = 5.01$  hingga  $5.99$ . Hal ini terus dilakukan hingga simulasi selesai.

*[Halaman ini sengaja dikosongkan]*



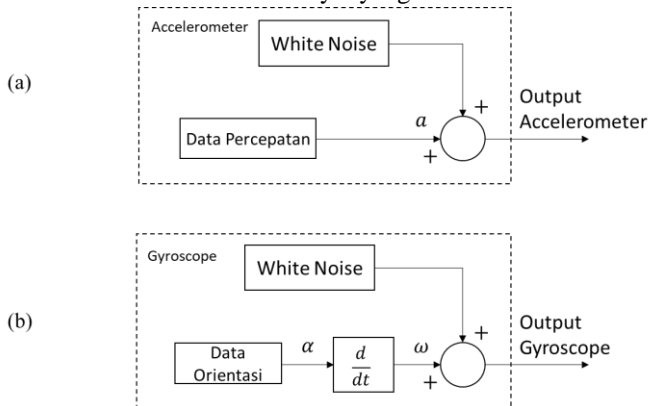
### BAB 3

#### PERANCANGAN DAN SIMULASI SISTEM

Perancangan sistem pada bab ini dibagi menjadi beberapa bagian, yaitu perancangan simulasi INS, perancangan simulasi GPS dan perancangan simulasi sistem terintegrasi.

#### 3.1 Perancangan Model INS

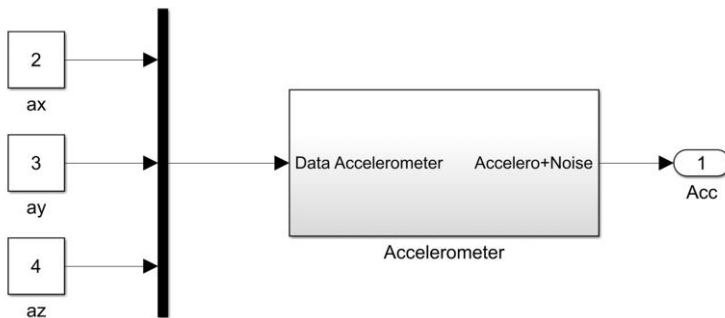
Perancangan simulasi INS pada subbab ini meliputi perancangan dari IMU yang berisi *accelerometer* dan *gyroscope*. Diagram blok yang menjadi dasar perancangan simulasi dapat dilihat pada gambar 3.1 (a) untuk perancangan simulasi *accelerometer* dan gambar 3.1 (b) untuk *gyroscope*. Sistem yang akan digunakan untuk simulasi terdiri atas IMU, filter LPF dan HPF untuk mendapatkan *error accelerometer* dan *gyroscope*, Model *error* dari INS dan Kalman filter sebagai estimator *error*. Pada implementasi praktisnya, *accelerometer* dan *gyroscope* dalam IMU membaca data percepatan dan kecepatan sudut. Data-data tersebut sudah mengandung *noise* di dalamnya yang diakibatkan oleh getaran mesin, efek gravitasi, efek koriolis, dan lain-lain, sehingga data keluaran dari *accelerometer* dan *gyroscope* adalah data yang sudah ditambah dengan *noise*. Seperti yang terlihat pada gambar 3.1, data percepatan dan kecepatan sudut ditambahkan dengan *noise* berupa *white noise*, sehingga keluaran dari sensor adalah data sinyal yang *bernoise*.



**Gambar 3.1** Diagram Blok *Accelerometer* (a) dan *Gyroscope* (b) INS

### 3.1.1 Perancangan Model *Accelerometer*

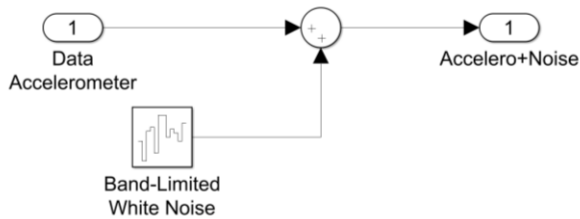
Seperti yang sudah dijelaskan pada subbab 3.1 mengenai perancangan simulasi INS, sensor pada IMU membaca data dan mengeluarkan data berupa sinyal yang telah ber-*noise*. Subsubbab ini lebih membahas tentang perancangan simulasi dari sensor *Accelerometer*. Karena sensor mengeluarkan data yang sudah ditambah dengan *noise*, penulis berasumsi bahwa penambahan *noise* terjadi pada sensor *accelerometer* seperti pada gambar 3.2, dimana input percepatan berupa nilai konstan masuk ke subsistem *Accelerometer* dan keluar sebagai sinyal ber-*noise*. Sinyal percepatan tersebut ditambahkan dengan *noise* dalam blok subsistem *Accelerometer* seperti pada gambar 3.3. *Noise* yang menyebabkan *error* pada *accelerometer* disebabkan oleh beberapa hal, seperti getaran pada *body* kendaraan karena getaran mesin atau karena medan yang dilalui, pengaruh percepatan gravitasi, dan lain-lain. *Noise* yang disebabkan oleh getaran biasanya terjadi dengan frekuensi tinggi. Karena itu, penulis mengasumsikan *noise* pada *accelerometer* adalah *noise* dengan frekuensi tinggi. Perlu diperhatikan bahwa pada implementasi praktisnya, asumsi ini bisa saja tidak berlaku/salah, dikarenakan *white noise* bekerja pada semua frekuensi. Untuk membangkitkan sinyal *white noise*, penulis menggunakan blok *band-limited white noise* yang ada pada simulink Matlab.



**Gambar 3.2** Diagram Simulink Perancangan *Accelerometer*

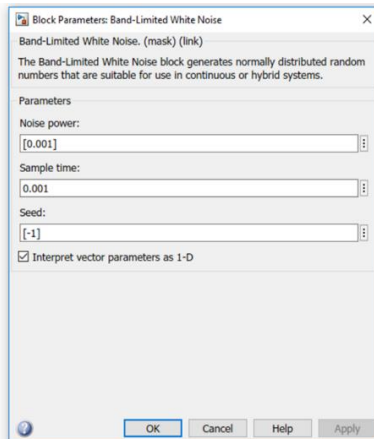
Sementara untuk nilai dari *noise* power dan time sampling dari *white noise* yang di-generate oleh blok *band-limited white noise* tersebut dapat dilihat pada gambar 3.4, nilai *time sampling* pada blok parameter tersebut

menandakan  $\frac{1}{f}$  dari frekuensi *noise* yang dihasilkan. Pada simulasi tugas akhir ini, penulis mengasumsikan *noise* yang terjadi pada *accelerometer* sebagai *noise* dengan frekuensi tinggi dengan  $f = 1000 \text{ Hz}$ , maka nilai dari *time sampling* yang dimasukkan adalah  $t_s = \frac{1}{f} = \frac{1}{1000} = 0.001 \text{ s}$ .



**Gambar 3.3** Blok Subsistem *Accelerometer* pada Simulasi INS

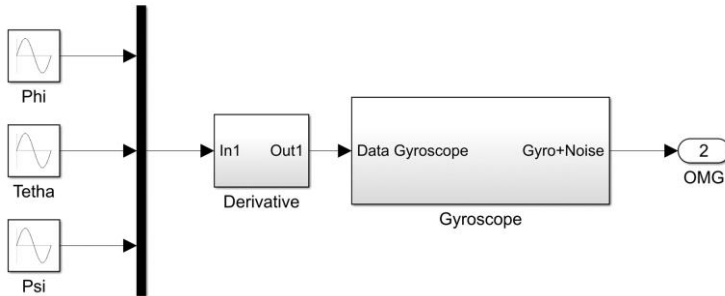
Nilai *white noise* yang di-generate oleh blok *Band-Limited White Noise* tersebut langsung ditambahkan ke data percepatan  $a_x$ ,  $a_y$ ,  $a_z$ . Data ini kemudian menuju blok LPF, menghasilkan data percepatan yang telah difilter. Nilai hasil filter tersebut akan dikurangkan dengan data percepatan sebelum difilter, menghasilkan estimasi *error accelerometer* untuk model *error*. Untuk penjelasan lebih lanjut akan dibahas pada subsubbab 3.1.3.



**Gambar 3.4** Blok Parameter dari *Band-Limited White Noise* pada *Accelerometer*

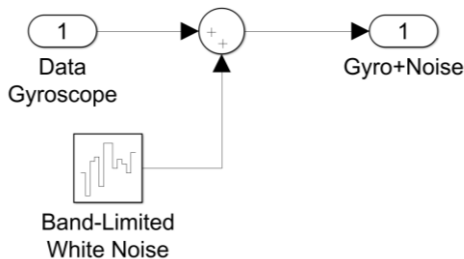
### 3.1.2 Perancangan Model Gyroscope

Perancangan *gyroscope* pada subbab ini meliputi model diagram simulink yang digunakan untuk simulasi dan merujuk pada gambar 3.1 (b) menghasilkan diagram simulink seperti pada gambar 3.5 berikut.



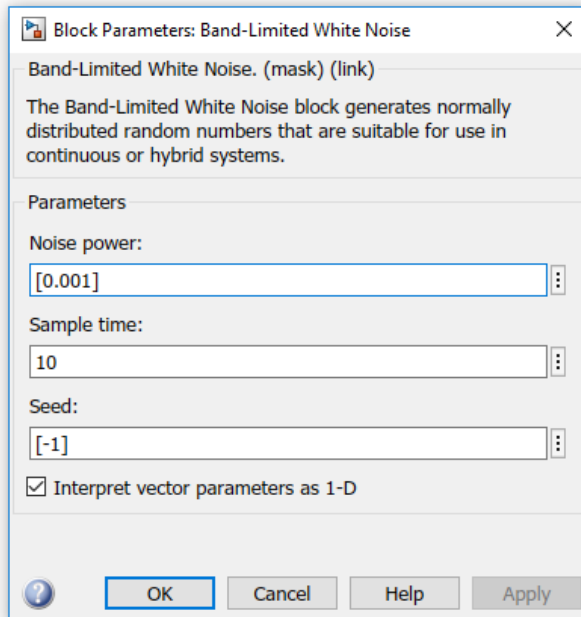
**Gambar 3.5** Blok Simulink Perancangan *Gyroscope*

Berdasarkan gambar 3.1 (b), *gyroscope* mendapatkan masukan berupa turunan dari orientasi, yakni kecepatan sudut dan menghasilkan keluaran berupa data kecepatan sudut ber-*noise*. Sama seperti perancangan pada *accelerometer*, *noise* pada *gyroscope* ditambahkan dalam blok subsistem *Gyroscope*. Untuk perancangan *Gyroscope*, data yang digunakan adalah orientasi benda, yakni sudut *roll*, *pitch*, *yaw*. Data orientasi tersebut diturunkan menjadi kecepatan sudut melalui blok *Derivative* seperti yang terlihat pada gambar 3.5. sedangkan gambar 3.6 menunjukkan bagian dalam dari blok subsistem *Gyroscope* pada simulink. Dimana nilai kecepatan sudut yang masuk ke subsistem *Gyroscope* ditambah dengan *white noise* yang di-*generate* dari blok *Band-Limited White Noise*.



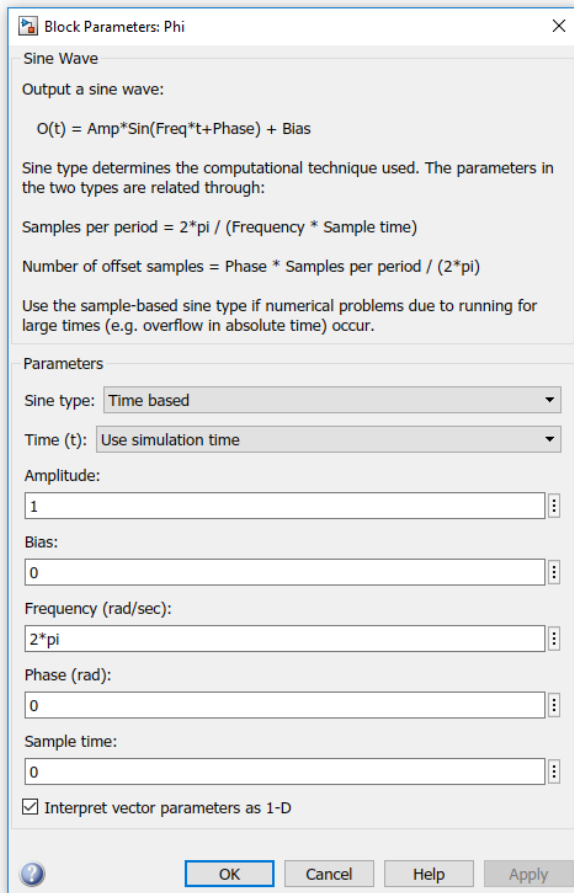
**Gambar 3.6** Blok subsistem *Gyroscope* pada INS

Untuk nilai frekuensi dari *white noise* yang dihasilkan, melihat salah satu dari hal yang menyebabkan *noise* pada *gyroscope* adalah efek dari rotasi bumi atau efek koriolis memiliki nilai yang sangat kecil untuk diamati (sekitar  $\pm 7.292115 \times 10^{-5}$  rad/s), maka penulis memilih 0.001 untuk *noise power* dan 0.1 Hz untuk frekuensi (*time sampling* = 10 s) dari *white noise* yang di-generate seperti pada gambar 3.7, dengan harapan *noise* tersebut mampu diamati.



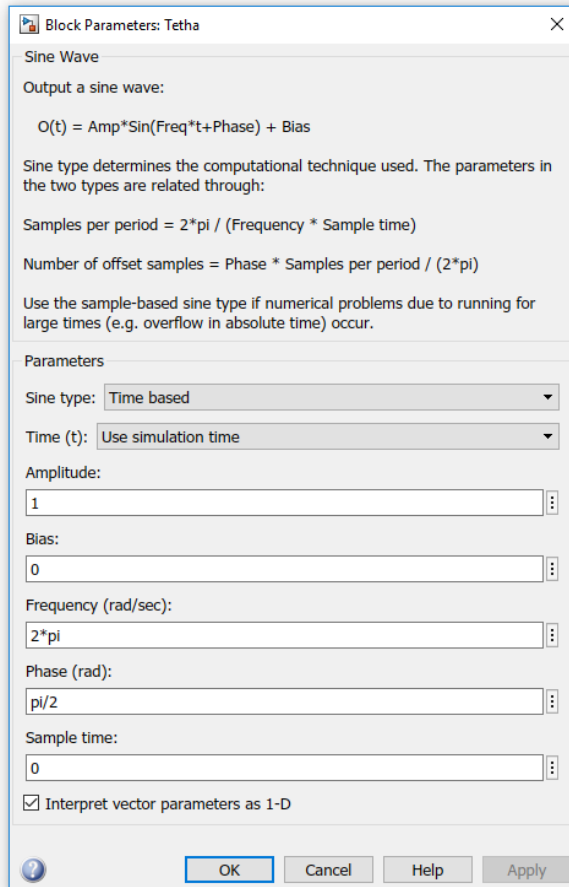
**Gambar 3.7** Blok Parameter dari *Band-Limited White Noise* pada *Gyroscope*

Sementara, untuk parameter dari input *gyroscope* berupa sudut *roll pitch* dan *yaw* dapat dilihat pada gambar 3.8 untuk *roll* ( $\Phi$ ,  $\varphi$ ), gambar 3.9 untuk *pitch* ( $\theta$ ,  $\theta$ ), dan gambar 3.10 untuk sudut *yaw* ( $\Psi$ ,  $\psi$ ).



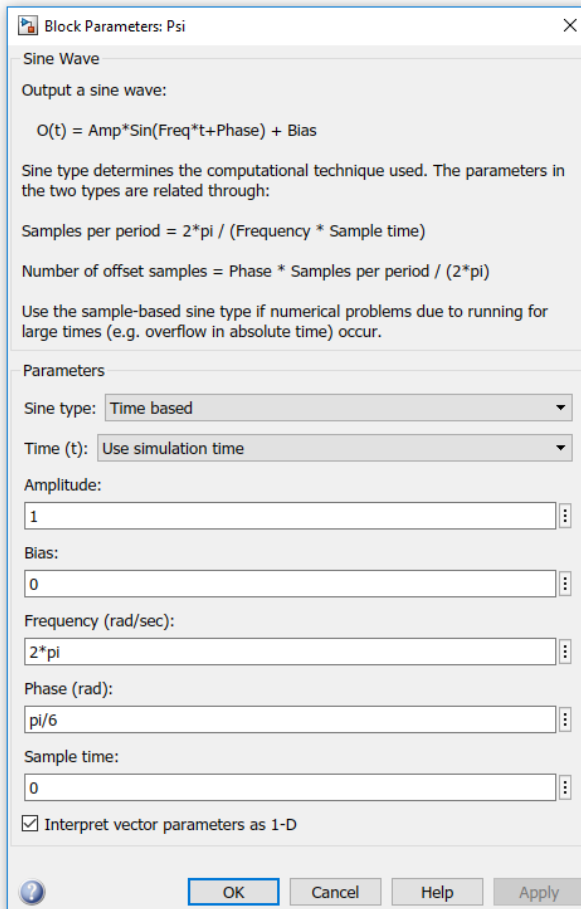
**Gambar 3.8** Blok parameter untuk data *roll* ( $\Phi$ ,  $\varphi$ )

Data sudut *roll* berupa sinyal sinus dengan amplitudo 1, frekuensi  $2\pi$  rad/s, dan sudut fasa 0.



**Gambar 3.9** Blok parameter untuk data *pitch* (theta,  $\theta$ )

Berdasarkan gambar 3.9 di atas, data sudut *pitch* terdiri atas sinyal sinus dengan amplitudo 1, frekuensi  $2\pi$  rad/s, dan sudut fasa  $\pi/2$  rad.



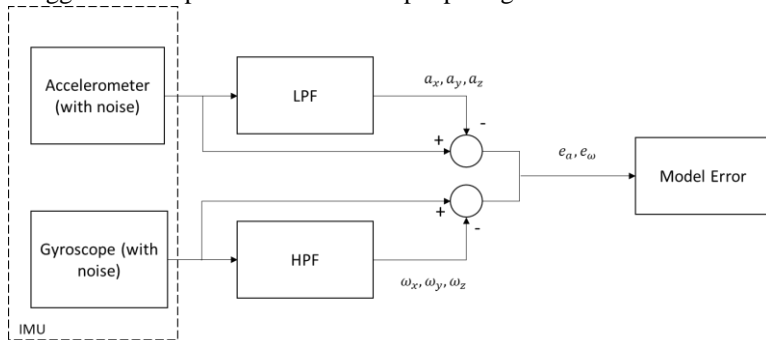
**Gambar 3.10** Blok parameter untuk data *yaw* (Psi,  $\psi$ )

Berdasarkan gambar 3.10 di atas, data sudut *yaw* dihasilkan dari blok sinyal sinus dengan amplitudo 1, frekuensi  $2\pi$  rad/s, dan sudut fasa  $\pi/6$ . Perlu diperhatikan bahwa data percepatan dan kecepatan sudut dalam tugas akhir ini merupakan asumsi yang tidak didasari pada data yang didapat dari pengukuran lapangan langsung dan digunakan semata-mata untuk keperluan simulasi.



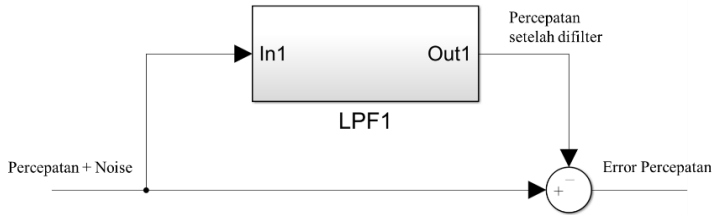
### 3.1.3 Perancangan LPF, HPF dan Model Error

Perancangan filter pada subsubbab ini adalah perancangan LPF dan HPF yang digunakan untuk memfiltrasi sinyal yang berisi *white noise*. Perlu diketahui, pada implementasi praktisnya, *white noise* bekerja pada setiap frekuensi, sehingga penggunaan LPF dan HPF tidak akan berpengaruh pada minimasi *noise*. Namun pada tugas akhir ini, simulasi dilakukan dengan blok *Band-Limited White Noise* yang men-generate *white noise* pada frekuensi tertentu, sehingga LPF dan HPF dapat digunakan untuk mendapatkan nilai *error accelerometer* dan *gyroscope* yang nantinya dapat digunakan untuk model *error*. Diagram blok untuk menggambarkan proses tersebut terdapat pada gambar 3.10 berikut ini

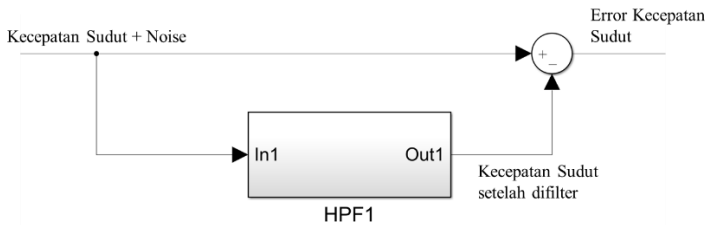


**Gambar 3.11** Diagram blok perancangan Model Error

Metode filtrasi yang digunakan adalah metode Butterworth, dengan pendekatan filter LPF dan HPF orde 2. Sinyal masukan berupa data percepatan *bernoise* dipecah menjadi tiga, masing-masing merupakan data pada setiap sumbu kartesian 3 dimensi (sumbu x, y, dan z) dan setiap data tersebut difilter oleh blok LPF seperti pada gambar 3.12. hasil keluaran dari LPF berupa data percepatan hasil filter dikeluarkan dari blok subsistem LPF (gambar 3.12). Hal yang serupa juga dilakukan untuk pengolahan data *gyroscope*, data *ber-noise* difilter oleh HPF menghasilkan keluaran berupa data kecepatan sudut hasil filter (gambar 3.13). Data-data hasil filter ini, dikurangkan dengan data *bernoise*, menghasilkan estimasi *noise* dari *accelerometer* dan *gyroscope* yang kemudian menjadi input/masukan bagi blok model *error* sebagai vektor *U* pada persamaan 2.53 dalam subbab 2.5.

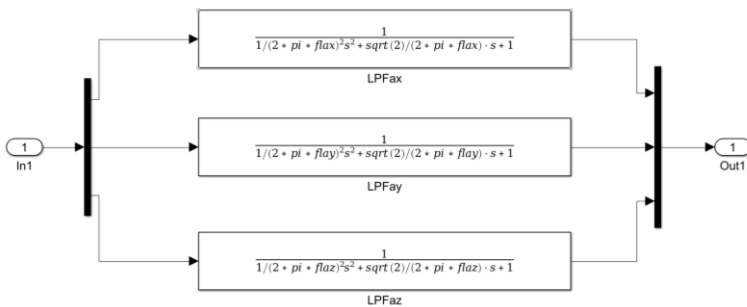


**Gambar 3.12** Diagram simulink untuk LPF

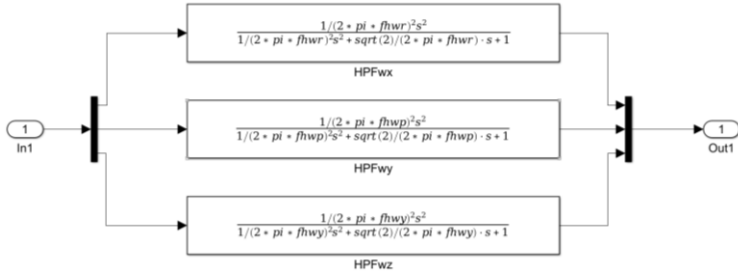


**Gambar 3.13** Diagram simulink untuk HPF

Untuk bagian dalam dari blok subsistem LPF dan HPF dapat dilihat pada gambar 3.14 dan gambar 3.15 berikut.



**Gambar 3.14** Bagian dalam blok Subsistem LPF



**Gambar 3.15** Bagian dalam blok subsistem HPF

Gambar 3.14 dan 3.15 merupakan blok transfer function yang dimasukkan persamaan LPF dan HPF, masing-masing untuk data pada sumbu x, y, dan z. Persamaan untuk LPF dan HPF tersebut adalah sebagai berikut.

$$H(s) = \frac{1}{\frac{1}{(2\pi\omega_c)^2} s^2 + \frac{\sqrt{2}}{2\pi\omega_c} s + 1} \quad (3.1)$$

Dan untuk HPF adalah:

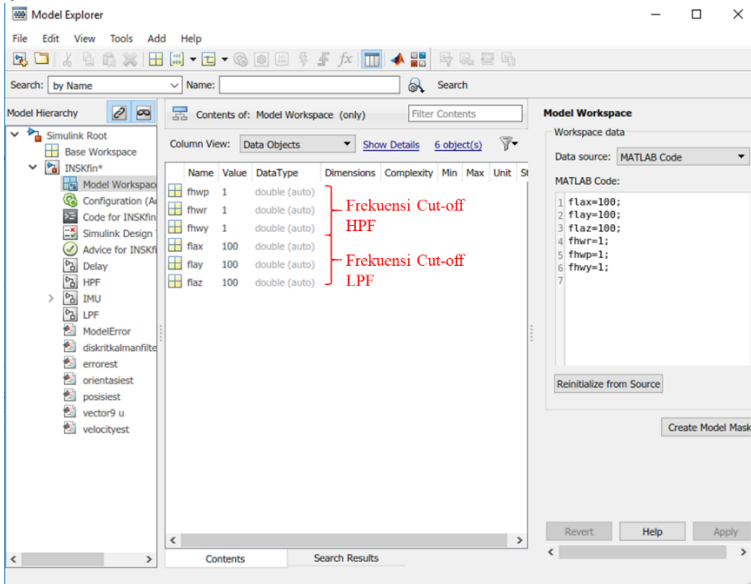
$$H(s) = \frac{\frac{1}{(2\pi\omega_c)^2} s^2}{\frac{1}{(2\pi\omega_c)^2} s^2 + \frac{\sqrt{2}}{2\pi\omega_c} s + 1} \quad (3.2)$$

Dimana  $\omega_c$  merupakan frekuensi *cut-off* dari LPF dan HPF. Berdasarkan dua persamaan di atas, filter yang digunakan adalah filter Butterworth orde 2. Sedangkan untuk nilai frekuensi *cut-off* dari LPF dan HPF masing-masing dimasukkan ke dalam model *workspace* pada *tab* MATLAB *Code*, dengan menuliskan variabel yang dimaksud dan nilainya. Gambar 3.16 adalah gambar dari model *workspace* yang berisi nilai frekuensi *cut-off* LPF dan HPF, sedangkan nilainya ada pada tabel 3.1 berikut

**Tabel 3.1** Nilai frekuensi *cut-off* dari LPF dan HPF

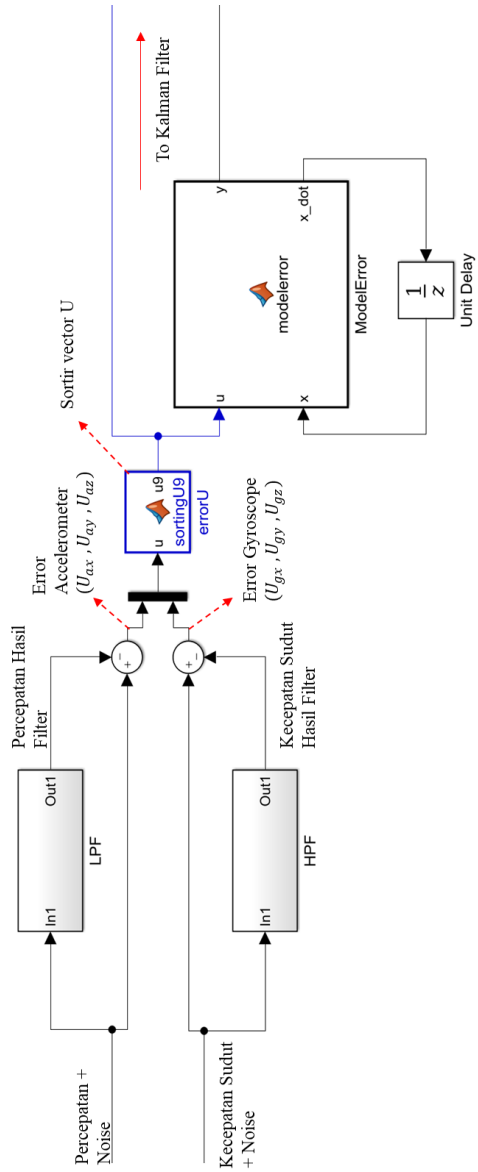
LPF	Nilai (Hz)
flax (frekuensi <i>cut-off</i> $\omega_x$ )	100
flay (frekuensi <i>cut-off</i> $\omega_y$ )	100
flaz (frekuensi <i>cut-off</i> $\omega_z$ )	100
HPF	Nilai (Hz)
fhwr (frekuensi <i>cut-off</i> $\omega_x$ )	1
fhwp (frekuensi <i>cut-off</i> $\omega_y$ )	1
fhwy (frekuensi <i>cut-off</i> $\omega_z$ )	1

Pada tabel 3.1 di atas, flax menandakan frekuensi *cut-off* dari LPF yang memfilter data percepatan sumbu-x ( $a_y$ ), sementara flay untuk  $a_y$  dan flaz untuk  $a_z$ . Sedangkan fhwr merupakan variabel frekuensi *cut-off* dari HPF yang memfilter data kecepatan sudut sumbu-x ( $\omega_x/roll$ ), fhwp untuk  $\omega_y/pitch$ , dan fhwy untuk  $\omega_z/yaw$ .



**Gambar 3.16** Nilai frekuensi *cut-off* dari LPF dan HPF

Setelah didapatkan *error* dari *accelerometer* dan *gyroscope*, nilai *error* tersebut digunakan sebagai input untuk mendapatkan model *error*. Gambar 3.17 menunjukkan diagram simulink dari proses mendapatkan model *error* berdasarkan diagram blok yang ada pada gambar 3.11

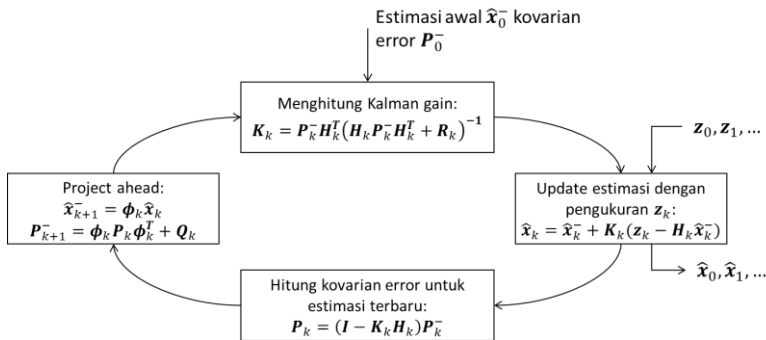


**Gambar 3.17** Diagram simulink dari model *error*

*Error* dari *accelerometer* dan *gyroscope* disortir oleh blok User-Identified Function yang berisi kode untuk mensortir masukan berupa data *error accelerometer* dan *gyroscope* sehingga menjadi vektor  $U$  seperti pada persamaan 2.53 pada subbab 2.5 mengenai model *error*. Vektor  $U$  tersebut digunakan sebagai input pada model *error* untuk mendapatkan data pengukuran *error* untuk simulasi. Vektor  $U$  yang berisi *error accelerometer* dan *gyroscope* tersebut juga dijadikan sebagai masukan pada blok Kalman filter untuk melakukan estimasi.

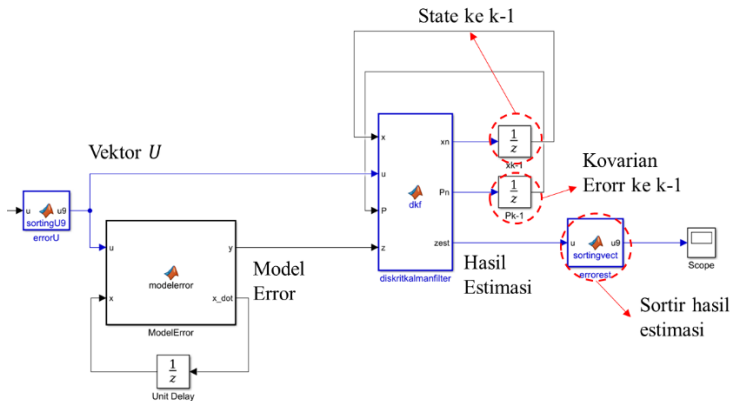
### 3.1.4 Perancangan Kalman Filter

Perancangan blok Kalman filter diawali dengan algoritma proses estimasi Kalman filter seperti yang dijelaskan pada subbab 2.3 dan ditunjukkan pada gambar 3.18 berikut



Gambar 3.18 Diagram blok proses estimasi filter Kalman

Setelah mendapatkan *error* dari *accelerometer* dan *gyroscope* dalam bentuk model *error*, dilakukan perancangan blok kalman filter untuk keperluan simulasi. Berdasarkan pada gambar 3.18, diketahui bahwa setiap satu siklus estimasi, filter kalman mengambil nilai estimasi  $x_{k-1}$  dan kovarian error  $P_{k-1}$ , yakni state estimasi dan kovarian error dari siklus yang lalu. Oleh karenanya, blok diagram simulink untuk Kalman filter memiliki tiga buah *output*, yakni hasil estimasi akhir, *output* estimasi awal, dan *output* kovarian error. Keluaran berupa estimasi awal dan kovarian error di-delay dan di-feedback menjadi *input*/masukan bagi Kalman filter untuk proses selanjutnya. Gambar 3.19 menunjukkan diagram simulink dari Kalman filter yang akan digunakan untuk simulasi.



**Gambar 3.19** Blok simulink Kalman filter

Kemudian, setelah dihasilkan keluaran berupa hasil estimasi *error* INS, hasil estimasi tersebut disortir ke dalam urutan estimasi *error* posisi, estimasi *error* kecepatan, dan estimasi *error* orientasi, masing-masing terhadap sumbu x, y, dan z secara berurutan.

Hasil dari estimasi *error* tersebut dikurangkan terhadap posisi, kecepatan dan orientasi yang mengandung *error*, yang didapatkan dari hasil integral dari data *accelerometer* dan *gyroscope* yang masing mengandung *noise*, sehingga didapatkan estimasi posisi, kecepatan dan orientasi sebagai solusi navigasi dari sistem INS. Kemudian, untuk membandingkan solusi navigasi yang didapatkan dari estimasi *error* filter Kalman dan mengetahui akurasi dari estimasi filter Kalman, solusi navigasi tersebut dibandingkan dengan data posisi, kecepatan dan orientasi yang bebas dari *noise*. Data bebas *noise* tersebut bisa didapatkan dengan mengintegrasikan data percepatan dan kecepatan sudut sebelum ditambahkan *noise*, atau sebelum masuk ke blok sensornya masing-masing. Untuk diagram blok dari simulasi INS secara keseluruhan dapat dilihat pada subbab 4.1 bersama dengan penjelasan dan analisis dari hasil simulasi.

Secara keseluruhan, semua perancangan yang dibahas pada subbab ini akan membentuk sebuah diagram simulink sistem INS dengan Kalman filter seperti pada gambar 3.20 berikut.

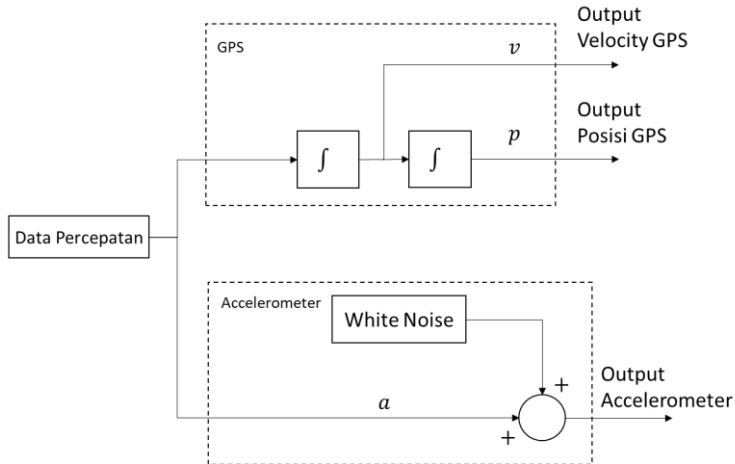
48



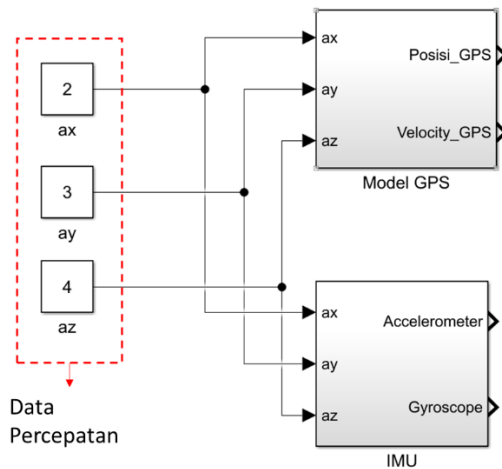
### 3.2 Perancangan Model GPS

Perancangan simulasi GPS hanya digunakan pada simulasi sistem terintegrasi INS-GPS, dan dibahas pada subbab 4.3. Pada implementasi praktisnya, GPS merupakan sistem navigasi yang masih memiliki *error* meski terbilang relatif lebih kecil. *Error* yang terjadi pada GPS diakibatkan oleh beberapa hal, antara lain kesalahan perhitungan *pseudorange*, *scattering*, jumlah satelit yang kurang dari 4 sehingga menghasilkan tracking yang tidak optimal, serta gedung-gedung tinggi yang mengganggu sinyal. Namun pada tugas akhir ini, data yang dihasilkan GPS dianggap bebas dari *noise* dan gangguan, dikarenakan GPS digunakan sebagai penyedia sinyal referensi bagi INS pada sistem terintegrasi.

Untuk memulai perancangan simulasi GPS, diasumsikan bahwa kedua sistem (INS dan GPS) berada pada satu *body* yang bergerak, bekerja pada waktu yang sama, sehingga akan menghasilkan data kecepatan dan posisi yang sama, hanya saja data keluaran dari INS akan mendapat *error* dari *noise*. Untuk itu, pada perancangan simulasi, INS dan GPS mendapat sumber data yang sama, yakni percepatan konstan seperti pada subsubbab 3.1.1, dimana percepatan tersebut diarahkan menuju dua blok, yakni blok IMU (menuju *accelerometer* untuk ditambahkan *noise*) dan blok GPS. Pada blok GPS, data percepatan kemudian diintegrasikan sebanyak dua kali tanpa ditambahkan *noise* apapun, sehingga keluaran dari blok GPS adalah data kecepatan dan posisi yang bebas dari *noise* sebagai sinyal referensi. Gambar 3.21 merupakan penggambaran dari diagram blok perancangan GPS yang akan digunakan dalam simulasi.

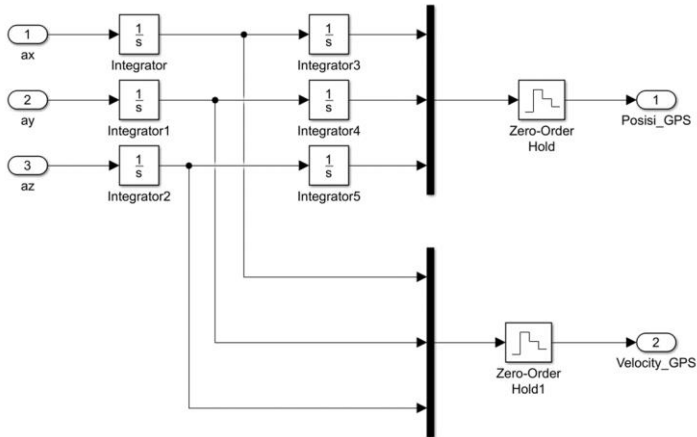


**Gambar 3.21** Diagram blok perancangan simulasi GPS  
Sedangkan gambar 3.22 merupakan blok simulink dari perancangan simulasi GPS.



**Gambar 3.22** Blok simulink perancangan GPS

Gambar 3.23 di bawah ini merupakan bagian dalam dari blok subsistem Model GPS yang ada pada gambar 3.22.



**Gambar 3.23** Bagian dalam blok Subsistem Model GPS

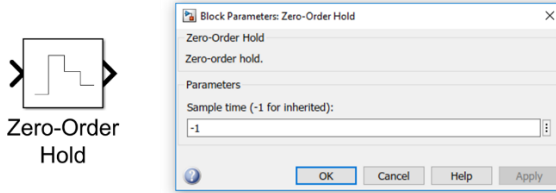
Berdasarkan gambar 3.23, blok subsistem Model GPS mendapat input berupa data percepatan. Berbeda dengan IMU, pada blok subsistem GPS data percepatan tersebut tidak ditambah dengan *noise*, melainkan hanya diintegrasikan dua kali untuk mendapatkan data keluaran berupa kecepatan dan posisi untuk setiap sumbu x, y, dan z. Sebelum data kecepatan dan posisi GPS dikeluarkan, data-data tersebut disampling dengan blok *Zero-Order Hold* dengan time sampling 1 detik, menghasilkan sinyal keluaran GPS dengan data rate 1 Hz.

Meski INS dan GPS bekerja pada waktu yang sama, namun data rate dari keduanya berbeda, data rate dari GPS lebih kecil dari INS. Untuk menyamakan/mensinkronkan *data rate* dari kedua sistem, dilakukan ekstrapolasi. Algoritma dari proses ekstrapolasi telah dijelaskan pada subbab 2.8 dan akan dijelaskan pada subsubbab berikutnya.

### 3.2.1 Perancangan Ekstrapolasi

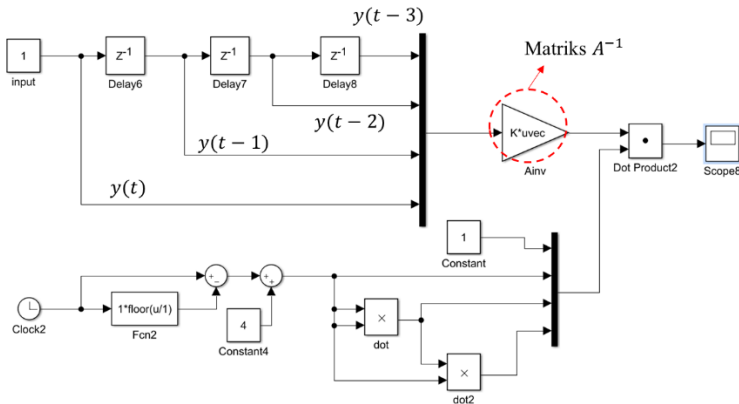
Seperti yang telah dijelaskan sebelumnya, bahwa INS dan GPS memiliki *data rate* yang berbeda. Untuk simulasi pada tugas akhir ini, data rate dari INS dipilih adalah 100 Hz dan data rate dari GPS dipilih 1 Hz. Untuk menimbulkan perbedaan data rate dari kedua sistem dalam simulasi menggunakan blok *Zero-Order Hold*. Blok ini dipasang setelah blok *accelerometer* dan GPS, dengan masukan berupa sinyal percepatan

bernoise (*accelerometer*) dan kecepatan serta posisi bebas noise (GPS). Untuk parameter data rate dari *Accelerometer*, diinginkan 100 Hz, maka time sampling yang perlu diisi pada blok *Zero-Order Hold* adalah 0.01, sementara *time sampling Zero-Order Hold* pada data GPS adalah 1 detik. Gambar 3.24 merupakan gambar blok *Zero-Order Hold* yang digunakan pada simulasi.



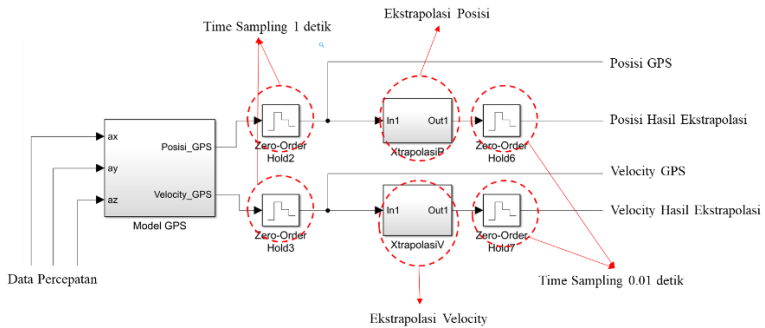
**Gambar 3.24** Blok simulink *Zero-Order Hold*

Sedangkan untuk merancang blok simulasi ekstrapolasi, berdasarkan persamaan 2.62 dan gambar 2.14, didapatkan blok simulink ekstrapolasi ditunjukkan oleh gambar 3.25 berikut



**Gambar 3.25** diagram simulink ekstrapolasi

Sementara penerapan diagram simulink ekstrapolasi tersebut dapat dilihat pada gambar 3.26 berikut.

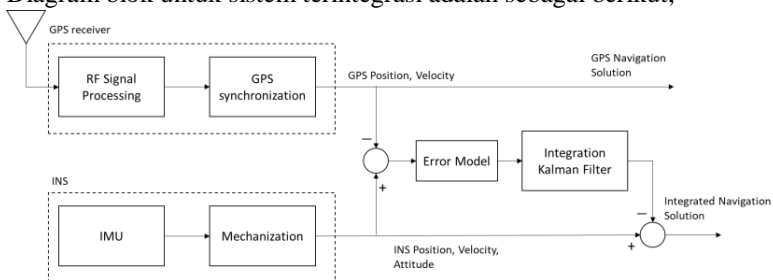


**Gambar 3.26** Diagram simulink Model GPS dan Ekstrapolasi

Pada gambar 3.25 tersebut keluaran dari GPS berupa data posisi dan kecepatan disampling dengan *time sampling* 1 detik, menghasilkan data GPS dengan *rate* 1 Hz. Data tersebut kemudian diekstrapolasi dalam blok Ekstrapolasi untuk setiap data kecepatan dan posisi dalam sumbu x, y, dan z untuk menghasilkan data kecepatan dan posisi yang memiliki *data rate* 100 Hz.

### 3.3 Perancangan Sistem Terintegrasi

Pada perancangan sistem terintegrasi, posisi dan kecepatan dari IMU dikurangkan dengan sinyal posisi dan kecepatan dari GPS untuk mendapatkan *error* posisi dan *velocity*. Persamaan model *error* untuk sistem terintegrasi adalah seperti pada persamaan (2.54) pada subbab 2.7. Diagram blok untuk sistem terintegrasi adalah sebagai berikut,



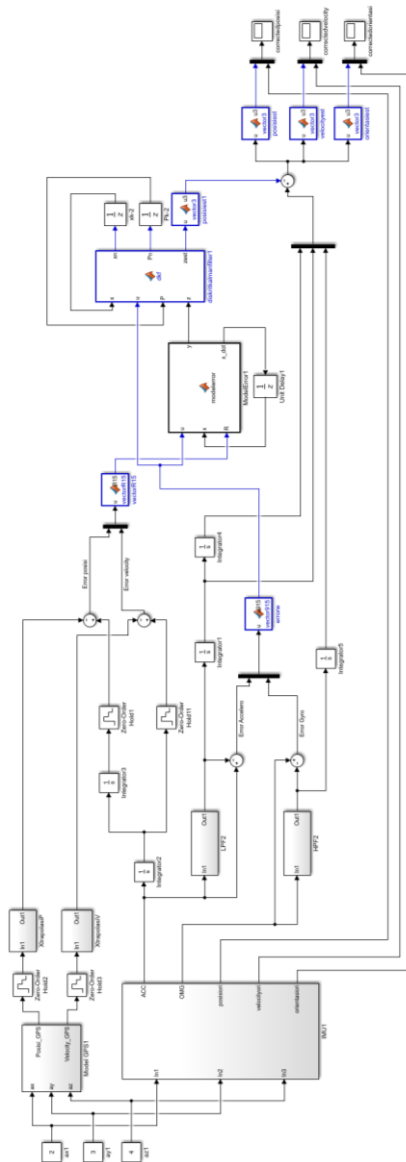
**Gambar 3.27** Diagram blok sistem terintegrasi INS-GPS *loosely coupled*

Gambar 3.27 di atas menjelaskan gambaran kasar dari sistem terintegrasi loosely-coupled. Data GPS berupa posisi dan kecepatan disinkronisasikan dengan data rate INS menggunakan algoritma

ekstrapolasi. Data GPS tersebut kemudian di kurangkan dengan data kecepatan dan posisi yang didapat dari turunan pertama dan kedua sinyal keluaran *accelerometer*. Hal ini akan menghasilkan estimasi *error* kecepatan dan posisi yang nantinya akan digunakan sebagai model *error*. Untuk data orientasi yang merupakan hasil turunan pertama dari kecepatan sudut data keluaran *gyroscope*, model *error*nya didapatkan dari hasil selisih antara keluaran HPF dengan orientasi yang masih *bernoise*. Ini akan menghasilkan estimasi *error* orientasi yang bisa digunakan untuk model *error*.

Sedangkan gambar 3.28 adalah diagram blok simulink yang digunakan dalam simulasi. Hasil simulasi dan analisa dari simulasi sistem terintegrasi INS-GPS akan dibahas pada subbab 4.3.

Terdapat perbedaan antara simulasi sistem INS stand alone dengan sistem terintegrasi. Selain daripada penambahan GPS sebagai sinyal referensi, model *error* dari sistem juga ikut berubah. Pada sistem terintegrasi, model *error* mendapat satu input tambahan, yakni *error* dari posisi dan kecepatan yang didapat dari selisih antara data GPS dengan INS. Persamaan untuk model *error* sistem terintegrasi seperti yang tertulis pada subbab 2.7. Untuk kode dari program yang digunakan, dapat dilihat pada lampiran.



**Gambar 3.28** Diagram blok simulink sistem terintegrasi INS-GPS

*[Halaman ini sengaja dikosongkan]*



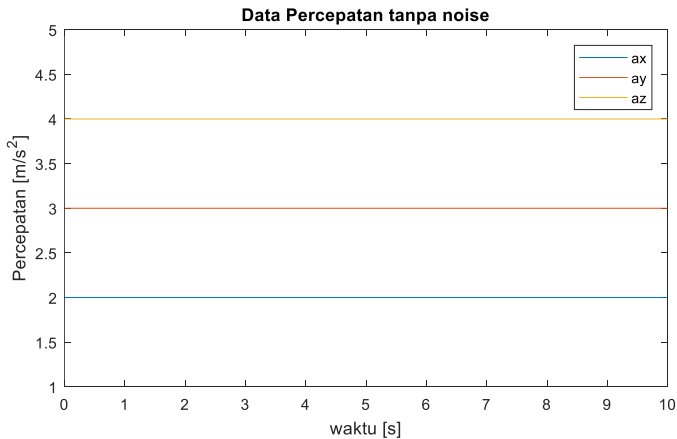
## BAB 4

### HASIL SIMULASI DAN ANALISA

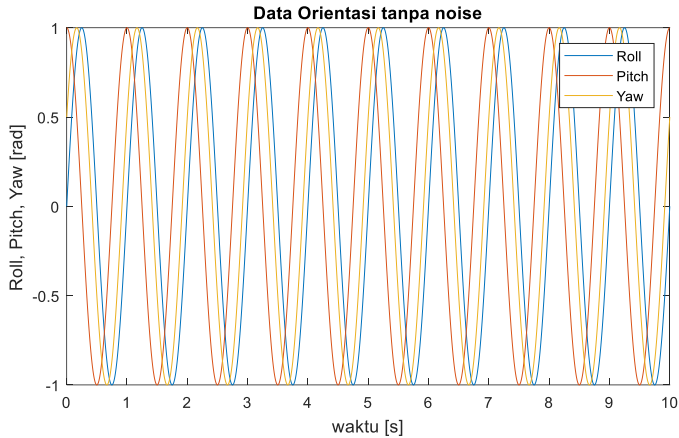
Pada bab ini, hasil simulasi dari perancangan simulasi yang dibahas di bab 3 akan ditunjukkan, beserta dengan analisis dari hasil simulasi yang dilakukan. Semua simulasi yang dilakukan dalam pengerjaan tugas akhir ini menggunakan perangkat lunak Matlab Simulink.

#### 4.1 Hasil Simulasi INS dengan Filter Kalman

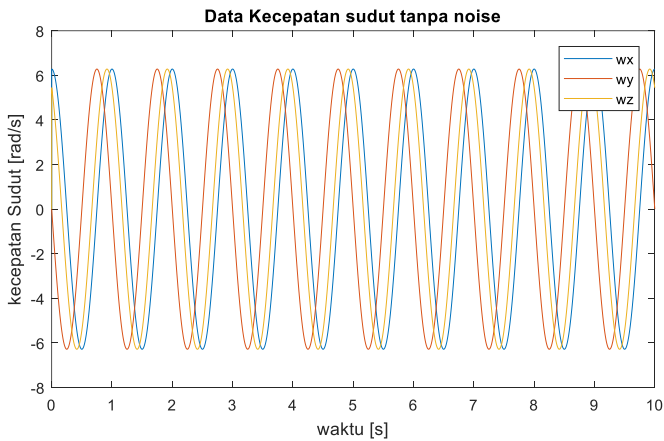
Subbab ini membahas tentang hasil simulasi dari sistem INS dengan Kalman filter yang telah dibahas dan dirancang pada subbab 3.1. Gambar 4.1 hingga 4.3 menunjukkan data percepatan, orientasi dan kecepatan sudut sebelum ditambahkan noise.



**Gambar 4.1** Data percepatan tanpa *noise*

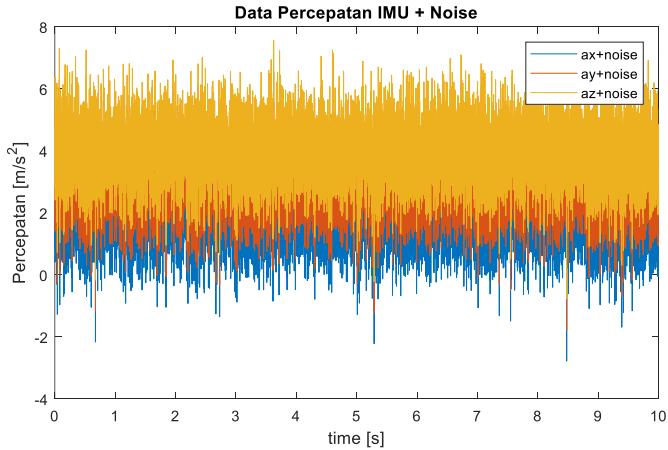


**Gambar 4.2** Data orientasi tanpa *noise*

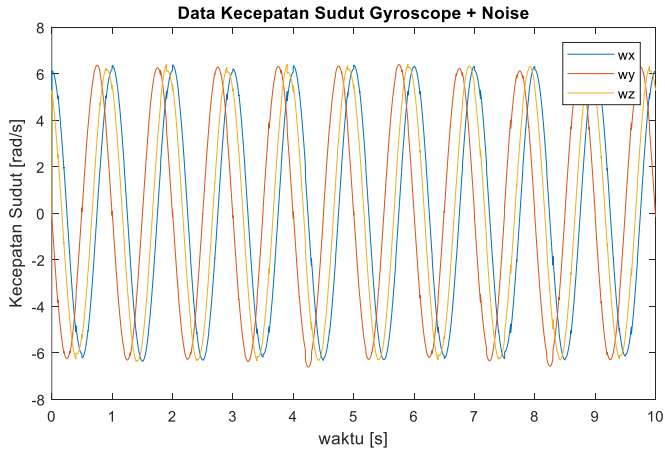


**Gambar 4.3** data kecepatan sudut tanpa *noise*

Data yang digunakan pada simulasi ini adalah data percepatan dan kecepatan sudut yang telah ditambah dengan *noise*, yakni gambar 4.4 untuk data percepatan ber-*noise* dan gambar 4.5 untuk kecepatan sudut ber-*noise*.

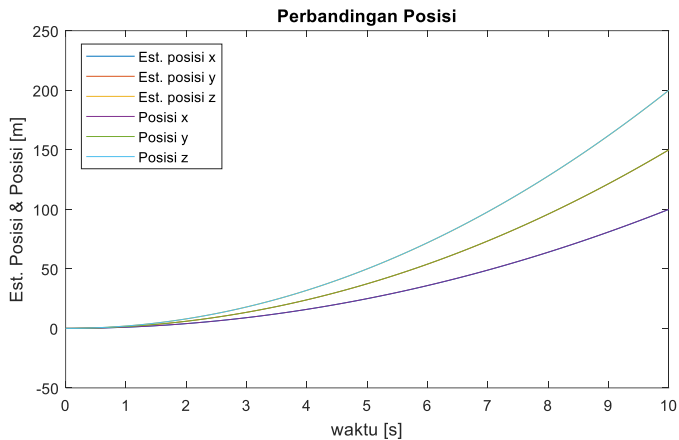


**Gambar 4.4** Data percepatan + *noise*

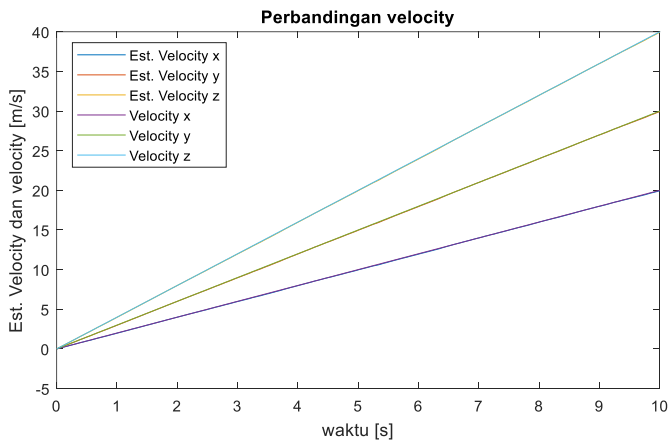


**Gambar 4.5** Data kecepatan sudut + *noise*

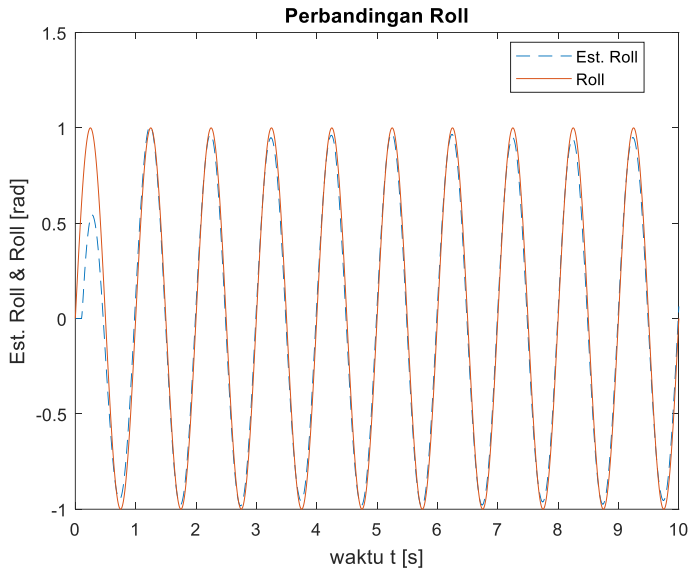
Sedangkan berikut adalah plot dari hasil antara estimasi filter kalman dengan data asli tanpa *noise* untuk posisi, kecepatan dan orientasi. Perbandingan antara data asli dengan hasil estimasi ini dilakukan untuk melihat dan mengetahui keakuratan kalman filter dalam estimasi *noise*.



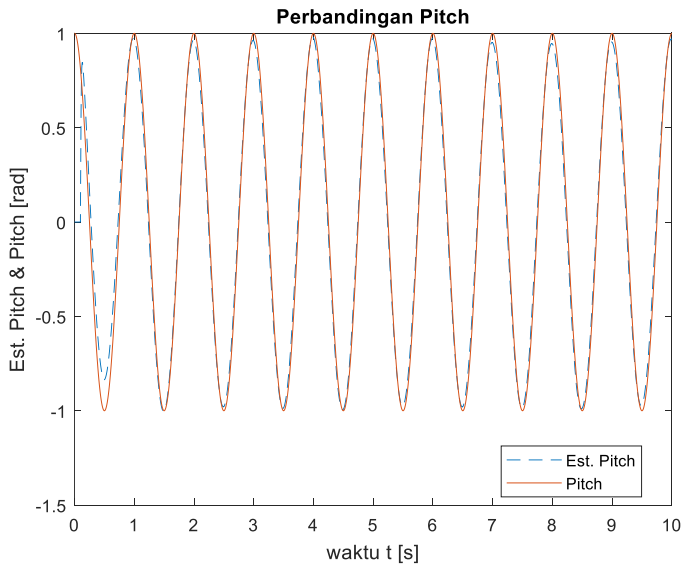
**Gambar 4.6** Perbandingan antara estimasi posisi dan posisi asli



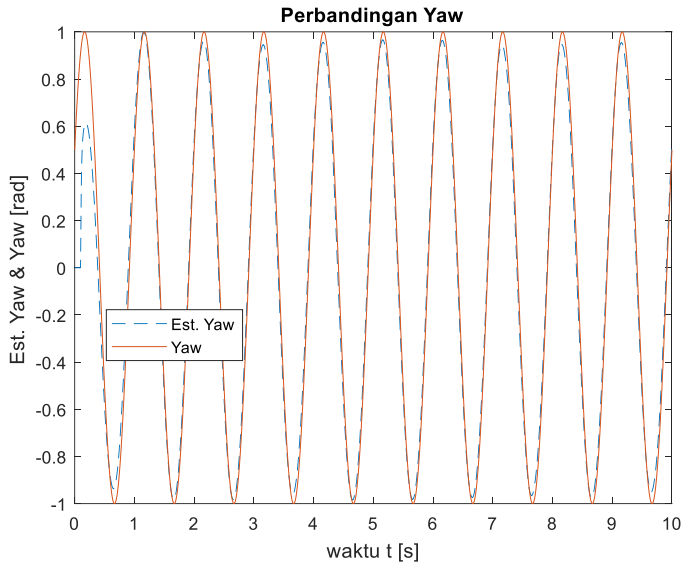
**Gambar 4.7** Perbandingan antara estimasi *velocity* dan *velocity* asli



**Gambar 4.8** Perbandingan sudut *roll*

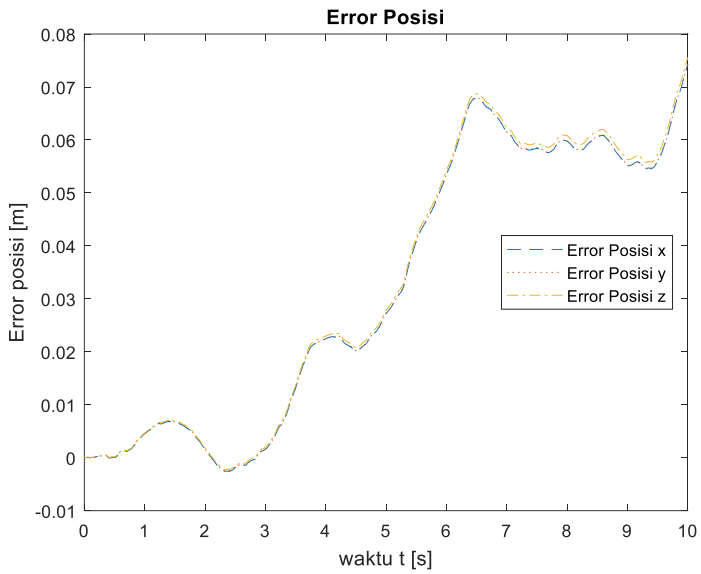


**Gambar 4.9** Perbandingan sudut *pitch*

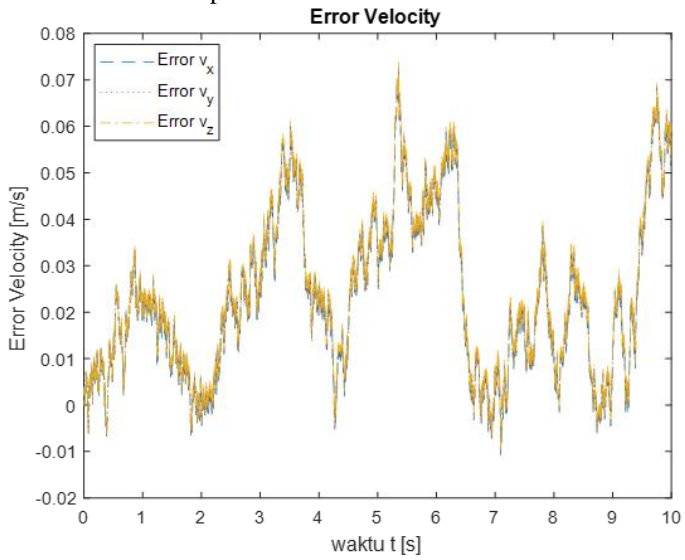


**Gambar 4.10** Perbandingan sudut yaw

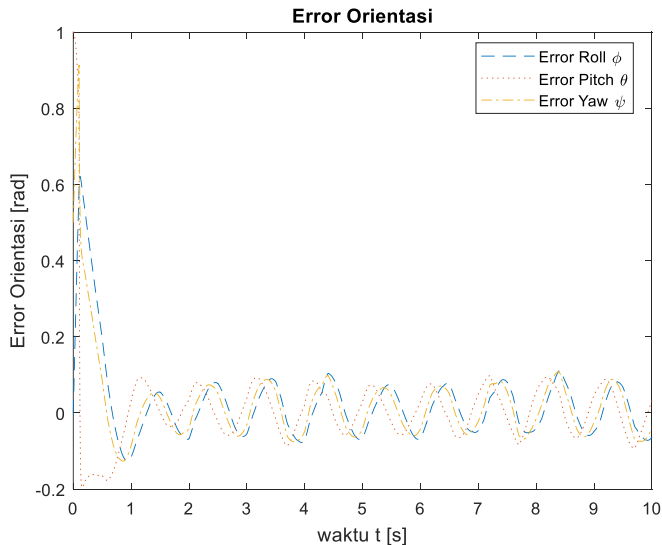
Dilihat secara sekilas, hasil perbandingan sinyal estimasi dengan sinyal asli tidak begitu berbeda. Karena itu, untuk mengetahui dengan lebih pasti apakah sinyal estimasi dari filter Kalman sudah sama dengan sinyal asli, maka dilakukan selisih antara sinyal asli dengan sinyal estimasi. Secara grafis, selisih antara sinyal asli dengan sinyal estimasi dapat dilihat pada gambar 4.11 hingga 4.13, sedangkan secara numerik dapat dilihat pada tabel 4.1.



**Gambar 4.11** *Error posisi*



**Gambar 4.12** *Error velocity*



**Gambar 4.13** Error orientasi

Tabel 4.1 memberikan informasi mengenai *error* dari hasil estimasi kalman filter dengan data asli tanpa *noise*. Data tersebut dibagi dalam nilai mean *error*, nilai maksimum *error*, nilai minimum *error* dan nilai varian dari *error* masing-masing untuk *error* posisi, kecepatan dan orientasi.

**Tabel 4.1.** Hasil statistik *error* untuk simulasi INS dengan Kalman filter

<i>Error Posisi</i> ( <i>Desired – Estimasi</i> )	[X, Y, Z]
<i>Mean</i>	[0.0323, 0.0323, 0.0329]
<i>Max</i>	[0.0743, 0.0744, 0.0756]
<i>Min</i>	[-0.0026, -0.0025, -0.0022]
<i>Varian</i>	[0.0258, 0.0258, 0.0261]
<i>Error Velocity</i> ( <i>Desired – Estimasi</i> )	
<i>Mean</i>	[0.0231, 0.0233, 0.0241]
<i>Max</i>	[0.0726, 0.0731, 0.0738]
<i>Min</i>	[-0.0108, -0.0103, -0.0096]
<i>Varian</i>	[0.0174, 0.0174, 0.0174]

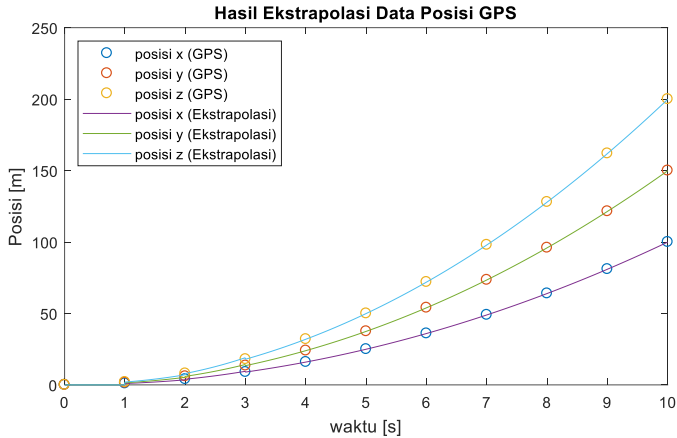


<i>Error Orientasi (Desired – Estimasi)</i>	
<i>Mean</i>	[0.0250, 0.0081, 0.0224]
<i>Max</i>	[0.6209, 1, 0.9135]
<i>Min</i>	[-0.1247, -0.1963, -0.1264]
<i>Varian</i>	[0.1073, 0.1173, 0.1104]

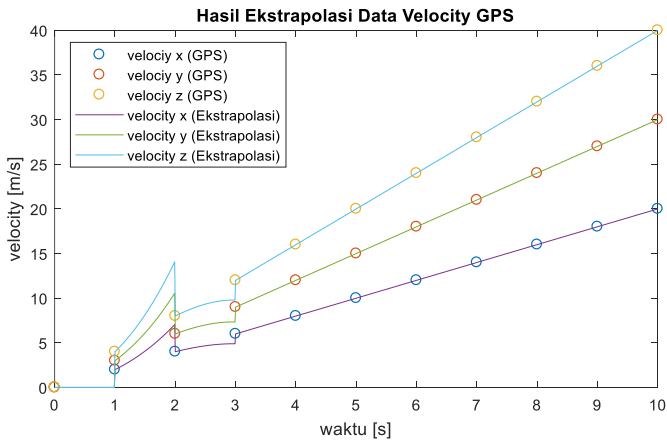
Berdasarkan Tabel 4.1, *mean error* antara sinyal asli/sinyal yang diinginkan dengan sinyal estimasi memiliki mean yang cukup kecil. Pada implementasinya, *White noise* yang merupakan aditif pada sinyal asli terjadi pada semua frekuensi. Hal ini menyebabkan LPF dan HPF tidak berguna untuk mendapatkan input model *error* yang diinginkan, lain hal jika kita mengetahui pada kisaran frekuensi berapa INS bekerja. Pada kasus seperti itu, kita dapat merancang sebuah *Band Pass Filter* (BPF) dengan frekuensi *cut-off* atas dan bawah berada di rentang frekuensi kerja INS. Karena hal tersebut sulit untuk dilakukan, maka digunakan GPS sebagai penyedia sinyal referensi untuk mendapatkan model *error* sistem terintegrasi.

## 4.2 Hasil Simulasi Ekstrapolasi Data GPS

Hasil simulasi yang dibahas pada subbab ini merupakan hasil dari simulasi perancangan model GPS dan perancangan ekstrapolasi yang telah dibahas dan dirancang pada subbab 3.2. Hasil dari plot antara data GPS asli, yakni data GPS berupa posisi dan kecepatan dengan data *rate* 1 Hz dengan hasil ekstrapolasi dengan *rate* 100 Hz ditunjukkan pada gambar 4.14 dan 4.15



**Gambar 4.14** Plot hasil ekstrapolasi data posisi GPS



**Gambar 4.15** Plot hasil ekstrapolasi data *velocity* GPS

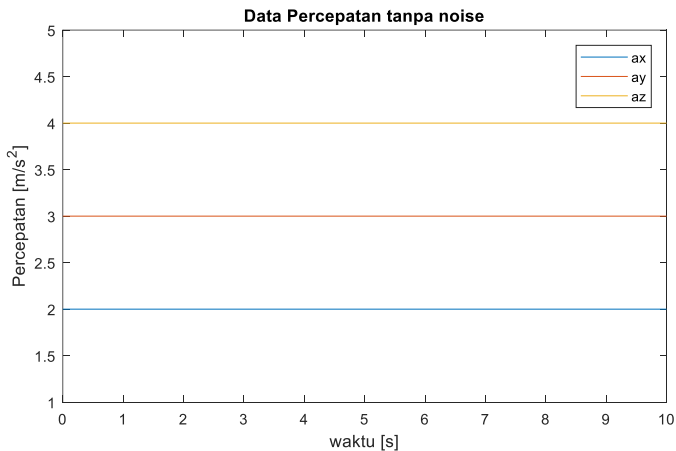
Berdasarkan gambar 4.14 dan 4.15 di atas, algoritma ekstrapolasi telah berhasil mengikuti sinyal asli. Pada plot hasil ekstrapolasi data posisi (gambar 4.14), ekstrapolasi berhasil dilakukan meski ada sedikit *error* pada , sedangkan pada plot hasil ekstrapolasi data kecepatan GPS (gambar 4.15), terdapat *error* pada ekstrapolasi dari detik ke 0 sampai ke 3. Hal ini dikarenakan belum adanya data yang mencukupi untuk menghitung parameter ekstrapolasi. Barulah setelah data keempat muncul,

pada detik ke-4, program ekstrapolasi berhasil mengikuti sinyal asli GPS dengan baik.

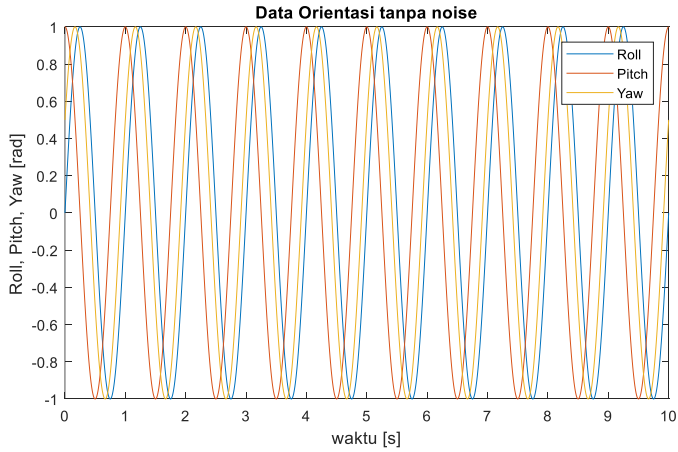
### 4.3 Hasil Simulasi Sistem Integrasi INS-GPS dengan Filter Kalman

Hasil simulasi pada subbab ini adalah berdasarkan perancangan sistem terintegrasi INS-GPS yang telah dibahas pada subbab 3.3.

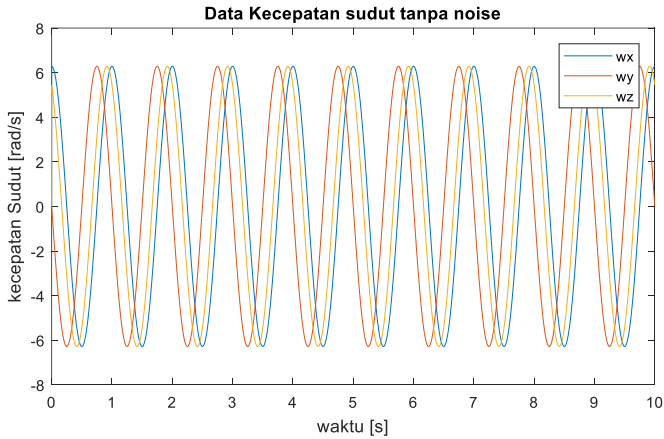
Untuk plot dari data yang digunakan, dapat dilihat pada gambar 4.16 hingga gambar 4.18, data *accelerometer* dan *gyroscope* ber-*noise* yang digunakan pada simulasi ditunjukkan pada gambar 4.19 dan gambar 4.20, Sementara gambar 4.21 dan gambar 4.22 merupakan data posisi dan kecepatan GPS yang merupakan sinyal referensi. Hasil estimasi Kalman filter yang merupakan hasil simulasi ditunjukkan pada gambar 4.23 hingga 4.25. sedangkan untuk error antara sinyal asli dengan hasil Kalman filter dapat dilihat pada gambar 4.26 hingga gambar 4.28.



**Gambar 4.16** Data percepatan tanpa *noise*

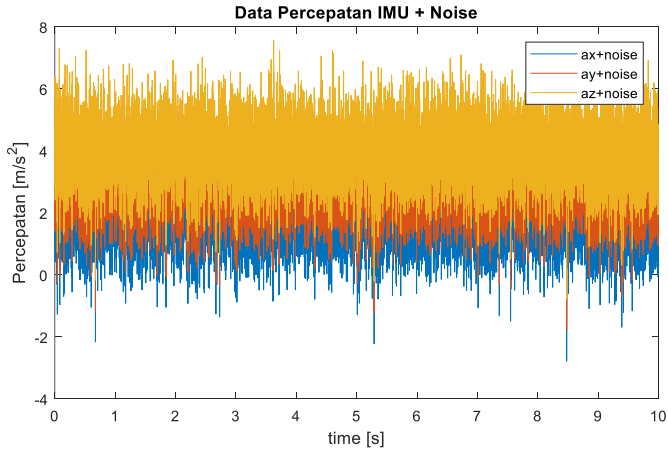


**Gambar 4.17** Data orientasi tanpa *noise*

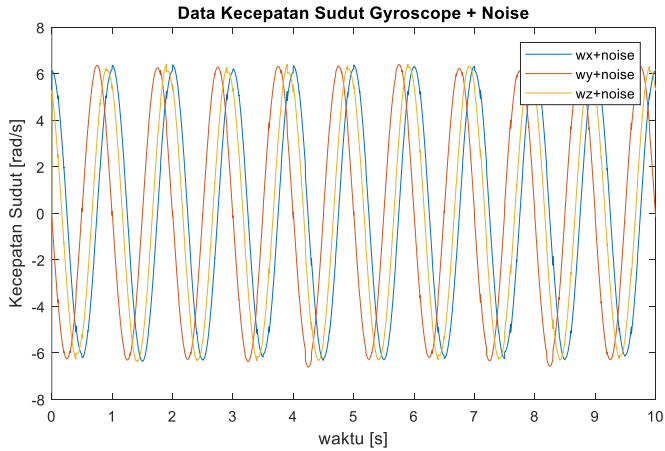


**Gambar 4.18** Data kecepatan sudut tanpa *noise*

Sedangkan gambar 4.19 dan 4.20 adalah plot dari data percepatan *accelerometer* dan *gyroscope* yang telah ditambah *noise*.

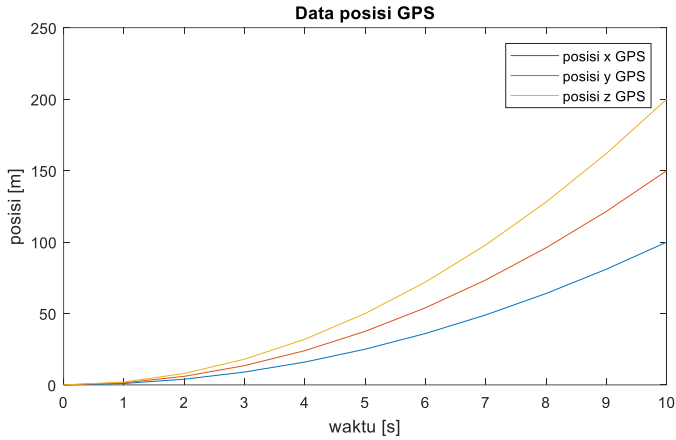


**Gambar 4.19** Data percepatan + *noise*

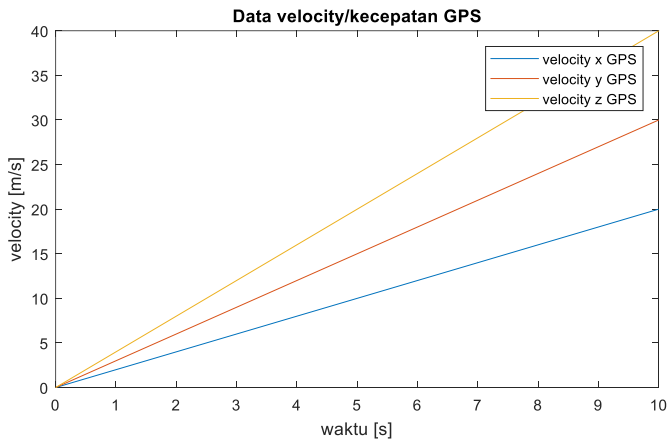


**Gambar 4.20** Data kecepatan sudut + *noise*

Untuk data kecepatan dan posisi dari GPS yang digunakan sebagai sinyal referensi pada simulasi ini, dapat dilihat pada gambar 4.21 untuk posisi dan gambar 4.22 untuk kecepatan.

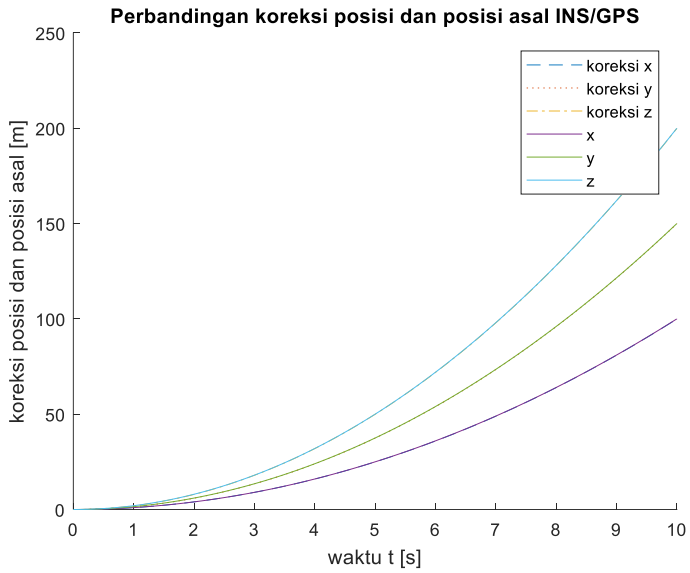


**Gambar 4.21** Data posisi GPS

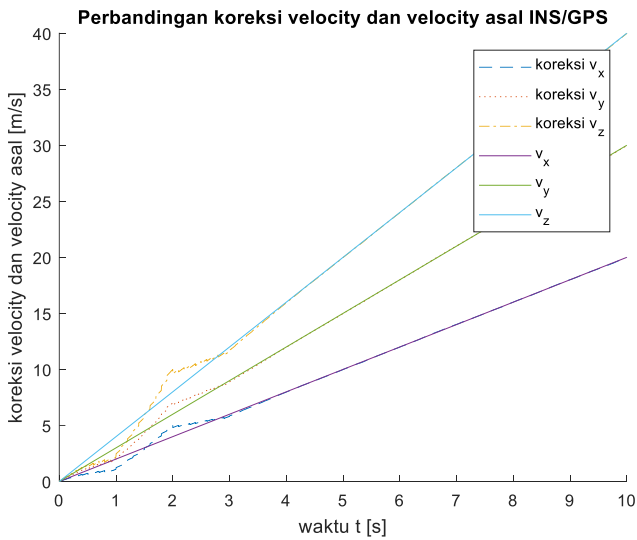


**Gambar 4.22** Data *velocity* GPS

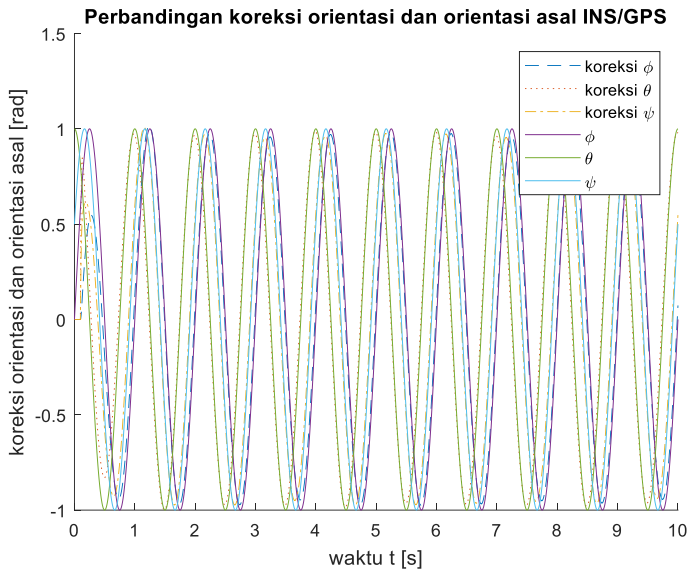
Gambar 4.23 hingga gambar 4.25 merupakan perbandingan antara koreksi estimasi Kalman filter dengan sinyal posisi, kecepatan dan orientasi tanpa *noise*.



**Gambar 4.23** Perbandingan posisi



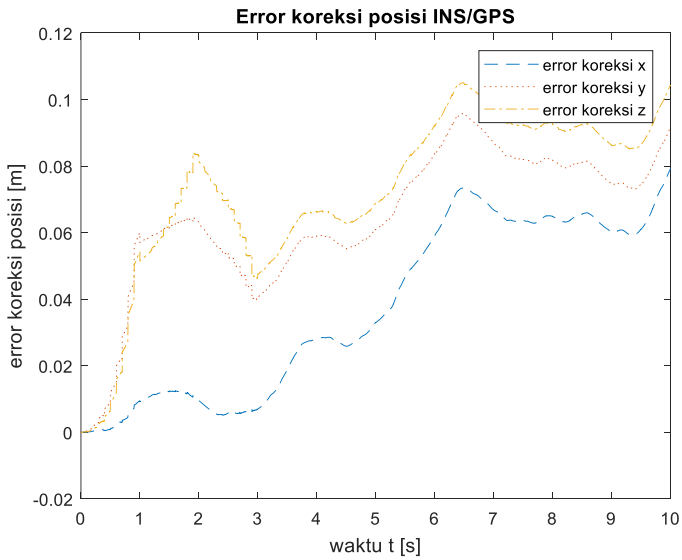
**Gambar 4.24** Perbandingan kecepatan (*velocity*)



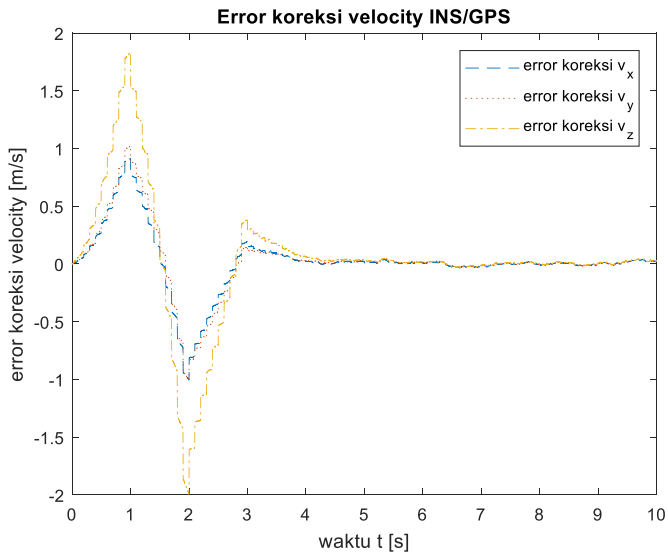
**Gambar 4.25** Perbandingan orientasi

Pada gambar 4.24, terjadi *error* yang cukup besar untuk  $t = 0$  hingga  $t = 4$ . Hal ini dikarenakan penyesuaian pendekatan polinomial orde 3 dari algoritma ekstrapolasi masih belum tercapai sepenuhnya. Barulah ketika waktu simulasi telah melebihi  $t = 4$  detik, sinyal estimasi berhasil mengikuti sinyal asli.

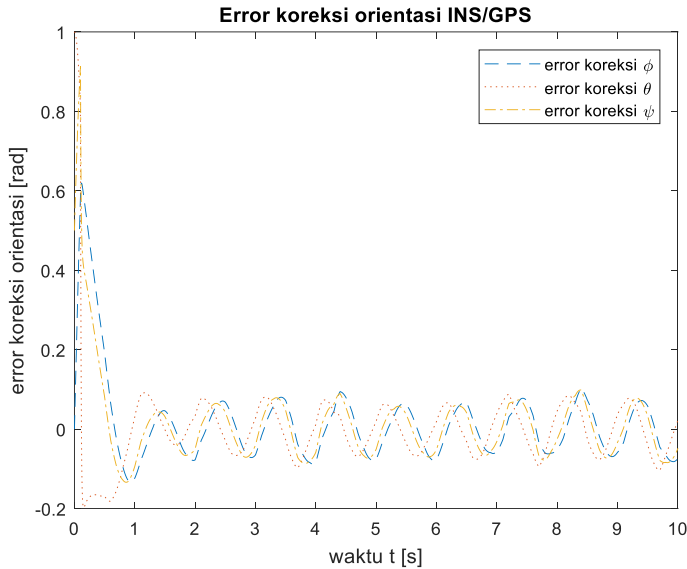




**Gambar 4.26** Error posisi



**Gambar 4.27** Error kecepatan



**Gambar 4.28** Error Orientasi

*Error* yang dikarenakan pendekatan polinomial algoritma ekstrapolasi terlihat lebih jelas pada gambar 4.27, yakni *error* kecepatan. Pada gambar tersebut, nilai *error* kecepatan pada detik ke-0 hingga detik ke-4 terlihat besar, barulah setelah detik ke-4, nilai *error* kecepatan mendekati nol. Sedangkan untuk membandingkan hasil dari sistem terintegrasi dengan simulasi INS dengan filter Kalman dilakukan dengan membandingkan hasil dari statistik *error* pada tabel 4.2 berikut.

**Tabel 4.2** Statistik *error* untuk hasil simulasi sistem INS-GPS terintegrasi

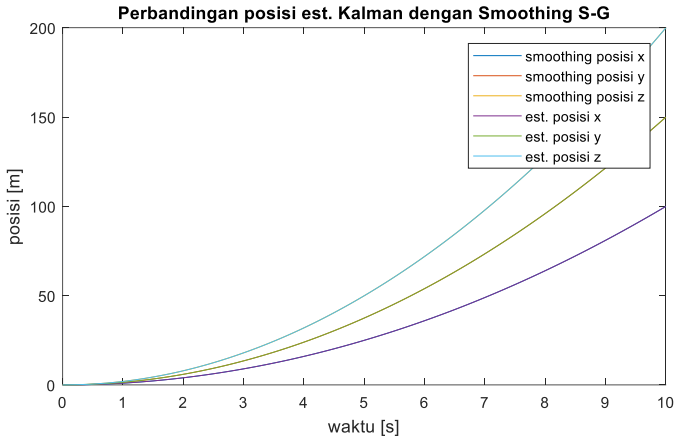
<i>Error Posisi</i> ( <i>Desired</i> – Estimasi)	[x, y, z]
<i>Min</i>	[-0.2687e-07, 0, -0.6597e-07]
<i>Max</i>	[0.0792, 0.0958, 0.1050]
<i>Mean</i>	[0.0375, 0.0644, 0.0724]
<i>Varian</i>	[0.0259, 0.0215, 0.0249]
<i>Error Velocity</i> ( <i>Desired</i> – Estimasi)	
<i>Min</i>	[-1.0056, -0.9996, -1.9937]

<i>Max</i>	[0.9164, 1.0210, 1.8325]
<i>Mean</i>	[0.0178, 0.0260, 0.0385]
<i>Varian</i>	[0.2678, 0.2906, 0.5326]
<i>Error Orientasi (Desired – Estimasi)</i>	
<i>Min</i>	[-0.1327, -0.1978, -0.1339]
<i>Max</i>	[0.6195, 1.0000, 0.9135]
<i>Mean</i>	[0.0157, -0.0007, 0.0132]
<i>Varian</i>	[0.1086, 0.1171, 0.1114]

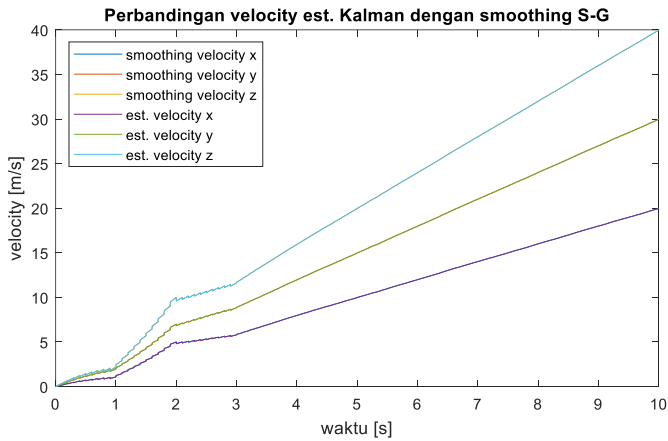
Jika dilihat dari mean *error* antara sinyal yang diinginkan dengan sinyal estimasi, hasil dari tabel 4.2 justru sedikit lebih tinggi dari mean *error* pada tabel 4.1. Hal ini diakibatkan oleh *error* yang muncul saat ekstrapolasi yang terjadi dari detik ke  $t = 0$  hingga  $t = 4$ . *Error* yang diakibatkan oleh proses ekstrapolasi ini menghasilkan *error* estimasi yang lebih tinggi dari simulasi sistem INS dengan Kalman filter (subbab 4.1).

#### 4.4 Hasil Simulasi Integrasi INS-GPS dengan Filter Kalman dan *Smoothing Savitzky-Golay*

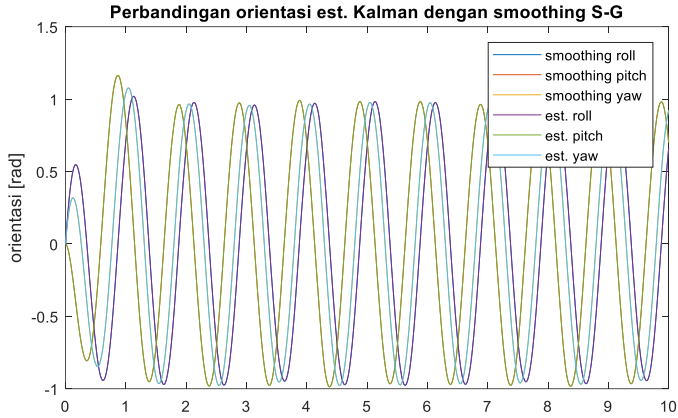
Untuk simulasi pada bab ini menggunakan function pada pernakat lunak Matlab yaitu `sgolayfilt(x, N, L)`, dengan  $x$  adalah sumber data (*workspace* pada Matlab) yang merupakan hasil estimasi Kalman filter,  $N$  adalah orde dari pendekatan polinomial yang digunakan, dan  $L$  adalah lebar (*length*) data yang diambil dalam proses *smoothing* ini. Proses *smoothing* ini dilakukan dalam keadaan *offline*, yakni saat simulasi telah berakhir dan data keluaran filter Kalman sudah didapatkan. Untuk simulasi pertama, digunakan pendekatan polinomial orde 3 ( $N = 3$ ) dan lebar data 25 ( $L=25$ ), sedangkan untuk percobaan kedua menggunakan pendekatan polinomial orde 4 ( $N = 4$ ) dan lebar data 25 ( $L = 25$ ). Hasil plot untuk percobaan pertama dapat dilihat pada gambar 4.34 sampai gambar 4.36, sedangkan untuk hasil plot percobaan kedua dapat dilihat pada gambar 4.37 hingga gambar 4.39 berikut.



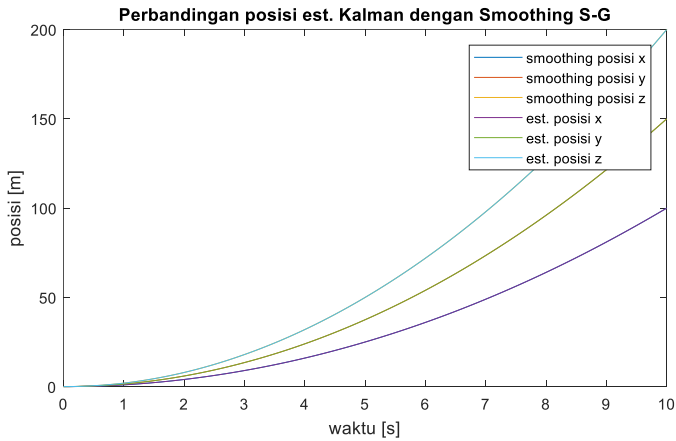
**Gambar 4.29** Perbandingan posisi antara est. Kalman dengan *smoothing* S-G untuk percobaan 1.



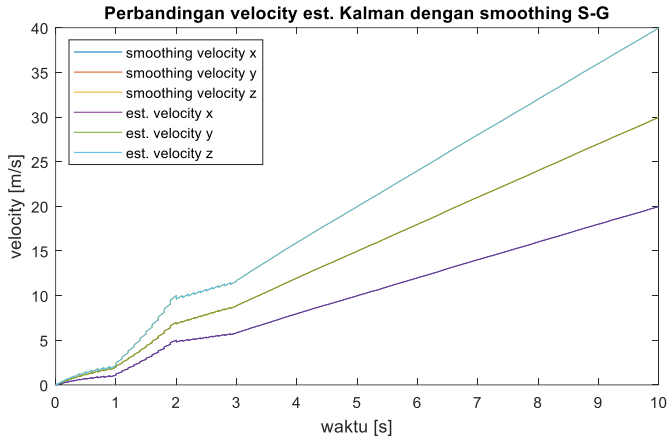
**Gambar 4.30** Perbandingan *velocity* antara est. Kalman dengan *smoothing* S-G untuk percobaan 1.



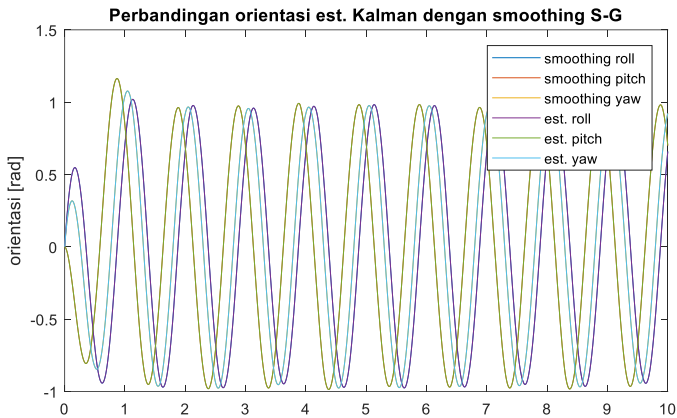
**Gambar 4.31** Perbandingan orientasi antara est. Kalman dengan *smoothing* S-G untuk percobaan 1.



**Gambar 4.32** Perbandingan posisi antara est. Kalman dengan *smoothing* S-G untuk percobaan 2.



**Gambar 4.33** Perbandingan *velocity* antara est. Kalman dengan *smoothing* S-G untuk percobaan 2.



**Gambar 4.34** Perbandingan *orientasi* antara est. Kalman dengan *smoothing* S-G untuk percobaan 2

Hasil perbandingan antara estimasi Kalman dengan algoritma *smoothing* Savitzky-Golay dapat dilihat pada tabel 4.3.

**Tabel 4.3** Perbandingan *error* statistik antara estimasi Kalman dan *smoothing* Savitzky-Golay.

<i>Error Posisi</i> ( <i>smoothing</i> – Kalman)	Percobaan 1 (N = 3 & L = 25)	Percobaan 2 (N = 4 & L = 25)
<i>Min</i>	[-0.0001, -0.0001, - 0.0016]	[-0.0001, -0.0001, -0.0013]
<i>Max</i>	[0.0001, 0.0001, 0.0014]	[0.0001, 0.0001, 0.0014]
<i>Mean</i>	[0.0098e-08, 0.100e-08, - 0.4059e-08]	[0.0031e-08, 0.0032e-08, 0.1769e-08]
<i>Varian</i>	[0.0015e-07, 0.0009e-07, 0.0012e-7]	[0.0011e-07, 0.0007e-07, 0.3428e-07]
<i>Error Velocity</i> ( <i>Smoothing</i> – Kalman )		
<i>Min</i>	[-0.0012, -0.0012, - 0.0012]	[-0.7448e-03, -0.7721e-03, - 0.7735e-03]
<i>Max</i>	[0.0012, 0.0010, 0.0010]	[-0.7576e-03, 0.8135e-03, 0.8395e-03]
<i>Mean</i>	[0.8383e-08, 0.8401e-08, 0.6683e-08]	[-0.2664e-08, -0.2673e-08, - 0.2810e-08]
<i>Varian</i>	[0.7857e-07, 0.7837e-07, 0.7860e-07]	[0.2188e-07, 0.2166e-07, 0.2169e-07]
<i>Error Orientasi</i> ( <i>smoothing</i> – Kalman)		
<i>Min</i>	[-0.0020, -0.0125, - 0.0062]	[-0.0016, -0.0075, -0.0038]
<i>Max</i>	[0.0017, 0.0132, 0.0066]	[0.0015, 0.0085, 0.0043]
<i>Mean</i>	[-0.0031e-06, -0.1485e-06, -0.0771e06]	[0.0101e-07, -0.8261e-07, - 0.4047e-07]
<i>Varian</i>	[0.0820e-06, 0.1068e-06, 0.0890e-06]	[0.6122e-07, 0.7445e-07, 0.6474e-07]

Pada tabel 4.3 di atas, mean *error* antara estimasi Kalman dengan *smoothing* sangat kecil, baik untuk data hasil *smoothing* menggunakan orde persamaan 3 dan lebar data 25 (percobaan pertama) serta orde persamaan 4 dan lebar data 25 (percobaan kedua). Hal ini menunjukkan bahwa hasil *smoothing* telah merhasil meminimasi *error* dari hasil estimasi filter Kalman, namun tidak memberikan perubahan yang signifikan pada data posisi, kecepatan dan orientasi yang merupakan solusi navigasi sistem INS-GPS. Melihat mean error antara percobaan satu dan percobaan dua, penulis berkesimpulan bahwa peningkatan orde

dari *smoothing* Savitzky-Golay memberikan pengaruh dalam minimasi error. Untuk implementasi di bidang navigasi, algoritma *smoothing* Savitzky-Golay telah mampu meningkatkan akurasi dari sistem INS-GPS untuk melakukan *tracking* rute perjalanan dengan baik, namun tidak memberikan pengaruh untuk implementasi INS di bidang kontrol kendaraan dan alat-alat bergerak lainnya.



## **BAB 5**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

Dari hasil simulasi dan analisa pada bab 4, penulis mengambil beberapa kesimpulan sebagai berikut:

1. Filter Kalman sudah mampu mengestimasi sinyal navigasi dengan baik, meski ada *error*, namun nilainya tidak terlalu besar. Hal ini dilihat berdasarkan nilai mean *error* dari *error* posisi [0.0323, 0.0323, 0.0329] m, *error* kecepatan/*velocity* [0.0231, 0.0233, 0.0241] m/s, dan *error* orientasi [0.025, 0.0081, 0.00224] rad, masing-masing dalam sumbu [x,y,z].
2. Algoritma ekstrapolasi kesulitan mengikuti sinyal asli GPS pada detik ke 0 hingga detik ke 3. Hal ini diakibatkan data yang ada belum mencukupi untuk mencari parameter polinomial, barulah ketika data GPS ke 4 muncul pada detik ke 4, algoritma ekstrapolasi berhasil menghitung parameter polinomial dengan baik karena data yang dibutuhkan telah mencukupi dan algoritma ekstrapolasi berhasil mengikuti sinyal GPS.
3. Melihat mean *error* pada tabel 4.3, penulis mengambil kesimpulan bahwa hasil antara estimasi Kalman dengan Algoritma *smoothing* Savitzky-Golay tidak berbeda jauh, meski nilai *error* berhasil di minimasi. Pada implementasi INS sebagai sistem navigasi, mungkin algoritma *smoothing* Savitzky-Golay mampu menghasilkan tracking dari rute perjalanan kendaraan dengan lebih baik.

#### **5.2 Saran**

Saran yang penulis berikan untuk pengembangan berikutnya dari tugas akhir ini adalah bahwa model simulasi yang dirancang pada tugas akhir ini masih sangat sederhana, belum ada elemen-elemen lain dalam topik navigasi seperti model gravitasi, transformasi frame, dan lain-lain. karenanya penulis berharap untuk kedepannya model sistem navigasi ini mampu dikembangkan hingga mendekati model aplikasi yang ada di lapangan, dan akan lebih baik jika diambil data lapangan untuk menguji sistem yang telah dirancang.

*[Halaman ini sengaja dikosongkan]*

### **DAFTAR PUSTAKA**

1. Abdoelmagd Noereldin, T. B. "Fundamentals of Inertial Navigation, Satellite-based Positioning and their Integration". Springer, Kingston, Canada, 2013
2. Groves, P. D. "Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems". Artech House, Boston, London, 2008
3. Mohinder S Grewal, A. P. "Global Navigation Satellite systems, Inertial Navigation, and Integration" (Thrid ed.), John Wiley & Sons, Danvers, 2013
4. Orfanidis, S. J. "Introduction to Signal Processing". Prentice Hall, 2010
5. Robert Grover Brown, P. Y. "Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises" (Fourth ed). John Wiley & Sons, Danvers, 2012
6. Anonymous, "Galileo (Satelite Navigation)", <[https://en.wikipedia.org/wiki/Galileo\\_\(satellite\\_navigation\)](https://en.wikipedia.org/wiki/Galileo_(satellite_navigation))>, Maret 2018

*[Halaman ini sengaja dikosongkan]*

## LAMPIRAN

Berikut ini adalah lampiran berupa program yang digunakan pada Tugas Akhir ini:

Lampiran 1: Model *Error* INS (*stand alone*)

```
%=====
% Model Error
%=====

function [y, x_dot] = modelerror(u, x)
deltat = 0.001;
    wx=-0.0000157;    %x-axis angular velocity
    [rad/s]
    wy= 0.0000727;    %y-axis angular velocity
    [rad/s]

    g = 9.8;          %gravitational acceleration
    [m/s^2]
    Re = 6380000;     %earth radius [m]

    A=[0  1  0  0  0  0  0  0  0;
        0  0  -g  0  0  0  0  0  0;
        0  1/Re  0  0  0  0  0  0  wx;
        0  0  0  0  1  0  0  0  0;
        0  0  0  0  0  -g  0  0  0;
        0  0  0  0  1/Re  0  0  0  wy;
        0  0  0  0  0  0  0  1  0;
        0  0  0  0  0  0  0  0  0;
        0  0  0  0  0  0  0  0  0]
    0]*deltat+eye(9);

    % B = eye(9)*deltat;
    Q = diag(eye(9)*[0 0.0001 0.0001 0 0.0001
0.0001 0 0.0001 0.0001])*deltat;
    % Q = diag(eye(9)*[0 0 0 1.4263 0.6339
0.3347 0.2458 0.2510 0.2466])*deltat;
    % Q = diag(eye(9)*[0 0 0 0.3 0.3 0.3 0.2458
0.2510 0.2466])*deltat;
```

```

% varaxyz =
%
%      1.4263      0.6339      0.3347
% varwrpy =
%
%      0.2458      0.2510      0.2466

% w = 0.01;
% x = [ex, evx, epi, ey, evy, etetha, ez,
evz, epsi]';
% x = [0 0 0 0 0 0 0 0 0]';

% x_dot = A*x+B*u+Q*w;
% x_dot = A*x+Q*u;
x_dot = A*x+Q*u;

C = eye(9);
R = eye(9)*[0.01 0.02 0.01 0.01 0.02 0.01
0.01 0.02 0.01]';%/deltat;
% nv = [0.01 0.02 0.01 0.01 0.02 0.01 0.01
0.02 0.01]';

% y = C*x+D*u+R*v;
y = C*x+R;
end

```

## Lampiran 2: Kalman Filter

```

%=====
% Kalman Filter
%=====
function [xn,Pn,zest] = dkf(x,u,P,z)
deltat = 0.001;
Q = diag(eye(9)*[0 0.0001 0.0001 0 0.0001
0.0001 0 0.0001 0.0001]')*deltat;
R = diag(eye(9)*[0.01 0.02 0.01 0.01 0.02
0.01 0.01 0.02 0.01]')*%/deltat;

```

```

    % nv = [0.01 0.02 0.01 0.01 0.02 0.01 0.01
0.02 0.01]';
    g=10;
    wx=-0.00002;
    wy=0.00007;
    Re=6380000;
    A=[0 1 0 0 0 0 0 0 0;
        0 0 -g 0 0 0 0 0 0;
        0 1/Re 0 0 0 0 0 0 wx;
        0 0 0 0 1 0 0 0 0;
        0 0 0 0 0 -g 0 0 0;
        0 0 0 0 1/Re 0 0 0 wy;
        0 0 0 0 0 0 0 1 0;
        0 0 0 0 0 0 0 0 0;
        0 0 0 0 0 0 0 0 0];
    0]*deltat+eye(9);
    %B=eye(9)*deltat;
    H=eye(9);

    % A=deltat*A+eye(9);
    % B=deltat*B;

    K=P*H'*inv(H*P*H'+R);
    x=x+K*(z-H*x);
    P=(eye(9)-K*H)*P;
    zest=H*x;

    xn=A*x;%+B*u;
    Pn=A*P*A'+Q;
end

```

### Lampiran 3: Model *Error* Sistem Terintegrasi INS-GPS

```

=====
% Model Error INS-GPS
=====
function [y, x_dot] = modelerror(u, x, R)
deltat = 0.001;

```

```

Q = diag(eye(15)*[0 0 0 0.2263 0.2339 0.2347
0.2458 0.2510 0.2466 0.2263 0.2339 0.2347 0.2458
0.2510 0.2466]')).*deltat;

```

```

g=10;
wx=-0.00002;
wy=0.00007;
Re=6380000;

```

```

A1=[0 1 0 0 0 0 0 0 0 0;
0 0 0 0 1 0 0 0 0 0;
0 0 0 0 0 0 0 1 0 0;
0 0 -g 0 0 0 0 0 0 0;
0 0 0 0 0 -g 0 0 0 0;
0 0 0 0 0 0 0 0 0 0;
0 1/Re 0 0 0 0 0 0 0 wx;
0 0 0 0 1/Re 0 0 0 0 wy;
0 0 0 0 0 0 0 0 0 0];

```

```

C1=[0 0 0 0 0 0 0;
1 0 0 0 0 0 0;
0 0 0 1 0 0 0;
0 0 0 0 0 0 0;
0 1 0 0 0 0 0;
0 0 0 0 1 0 0;
0 0 0 0 0 0 0;
0 0 1 0 0 0 0;
0 0 0 0 0 1 0];

```

```

pax=-0.0001;
pay=-0.0001;
paz=-0.0001;
pgx=-0.0001;
pgy=-0.0001;
pgz=-0.0001;
As=diag([pax,pay,paz,pgx,pgy,pgz]);

```

```

A=[A1 C1;zeros(6,9) As]*deltat+eye(15);
x_dot = A*x+Q*u;

```

```

C2 = [1 0 0 0 0 0 0 0 0 0;

```



```

        0 0 0 1 0 0 0 0 0;
        0 0 0 0 0 0 1 0 0;
        0 1 0 0 0 0 0 0 0;
        0 0 0 0 1 0 0 0 0;
        0 0 0 0 0 0 0 1 0];
    Cz = [C2 zeros(6,6)];
    Cz15=[Cz;zeros(9,15)];
    R15=[R; zeros(9,1)];
    y = Cz15*x+R15;
end

```

#### Lampiran 4: Savitzky-Golay *Smoothing*

```

%=====
% Savitzky-Golay
%=====
load('kalout.mat');
percobaan = 1;

switch percobaan
    case 1
        sm = sgolayfilt(KaloutN3(:,2:10),3,25);
    case 2
        sm = sgolayfilt(KaloutN3(:,2:10),4,25);
end
% plot posisi velocity n orientasi
figure(1);
plot(Kalout(:,1),sm(:,1:3),Kalout(:,1),Kalout(:,
2:4)); %posisi smooth n kalman
figure(2);
plot(Kalout(:,1),sm(:,4:6),Kalout(:,1),Kalout(:,
5:7)); %velocity smooth n kalman
figure(3);
plot(Kalout(:,1),sm(:,7:9),Kalout(:,1),Kalout(:,
8:10)); %orientasi smooth n kalman

% perhitungan error antara smoothing savitzky-
goley dengan kalman filter
errposisi = sm(:,1:3)-Kalout(:,2:4);
errvelocity = sm(:,4:6)-Kalout(:,5:7);

```

```

errororientasi = sm(:,7:9)-Kalout(:,8:10);

% min max mean varian posisi
minimumerrorposisi = min(errrposisi)
maximumerrorposisi = max(errrposisi)
meanerrorposisi    = mean(errrposisi)
varianterrorposisi = var(errrposisi)

% min max mean varian velocity
minimumerrorvelocity = min(errrvelocity)
maximumerrorvelocity = max(errrvelocity)
meanerrorvelocity    = mean(errrvelocity)
varianterrorvelocity = var(errrvelocity)

% min max mean varian orientasi
minimumerrororientasi = min(errororientasi)
maximumerrororientasi = max(errororientasi)
meanerrororientasi    = mean(errororientasi)
varianterrororientasi = var(errororientasi)

```

#### Lampiran 5: Sortir Error Model

```

%=====
% Sortir input Model Error
%=====
function u9 = vector9(u)
% This block supports the Embedded MATLAB
subset.
% See the help menu for details.
u9 = [0 u(1) u(4) 0 u(2) u(5) 0 u(3) u(6)]';
% u9 = [0 uax ugx 0 uay ugz]'

```

#### Lampiran 6: Sortir Hasil Estimasi Kalman filter

```

%=====
% Sortir Hasil Estimasi Kalman filter
%=====
function u9 = sortingvect(u)
% This block supports the Embedded MATLAB
subset.
% See the help menu for details.

```

```

        u9 = [u(1) u(4) u(7) u(2) u(5) u(8) u(3)
u(6) u(9)]';
        % u9 = [posisi-x posisi-y posisi-z velocity-
x velocity-y velocity-z
        % orientasi-x orientasi-y orientasi-z]'
end

```

*[Halaman ini sengaja dikosongkan]*

## **RIWAYAT HIDUP PENULIS**



Maulana Maliki lahir pada 11 September 1994 di kota Pasuruan, Jawa Timur, merupakan anak kedua dari dua bersaudara dari pasangan H. Margono (Alm.) dan Wiwik Handayani. Memiliki seorang kakak perempuan bernama Ranny Rufaidah yang 2 tahun lebih tua. Setelah menyelesaikan pendidikan sekolah dasar pada tahun 2007, sekolah menengah pada tahun 2010, dan sekolah menengah atas pada tahun 2013, penulis meneruskan pendidikannya di Institut Teknologi Sepuluh Nopember, Surabaya di departemen Teknik Elektro, Fakultas Teknik Elektro. Penulis memiliki minat pada perancangan/simulasi dan sistem membuat penulis memilih teknik Sistem pengaturan sebagai bidang studi. Hingga saat ini, penulis terus berusaha agar menjadi seorang yang bermanfaat bagi orang lain, bangsa dan negara.

*[Halaman ini Senaja dikosongkan]*